COMPUTER REPRESENTATION OF BUILDING CODES FOR AUTOMATED COMPLIANCE CHECKING

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in Architecture

by Sibel MACİT

November 2014
İzmir

We approve the thesis of **Sibel MACİT**

Examining Committee Members:

Prof. Dr. H. Murat GÜNAYDIN
Department of Architecture, İzmir Institute of Technology

Prof. Dr. Serdar KALE
Department of Architecture, İzmir Institute of Technology

Assoc. Prof. Dr. Georg SUTER
Institute of Architectural Sciences, Vienna University of Technology

---

Prof. Dr. Türkan GÖKSAL ÖZBALTA
Department of Civil Engineering, Ege University

Assoc. Prof. Dr. Koray KORKMAZ
Department of Architecture, İzmir Institute of Technology

19 November 2014

Prof. Dr. H. Murat GÜNAYDIN
Supervisor, Department of Architecture,
İzmir Institute of Technology

Assoc. Prof. Dr. Şeniz ÇIKIŞ
Head of the Department of
Architecture

Prof. Dr. R. Tuğrul SENGER
Dean of the Graduate School of
Engineering and Sciences

ABSTRACT

COMPUTER REPRESENTATION OF BUILDING CODES FOR AUTOMATED COMPLIANCE CHECKING

This dissertation constitutes a study in the field of automated compliance checking, with a concentration on building code representations. Development of 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. Systems for compliance checking of building projects require appropriate representations for building codes. Building codes are complex documents written in natural languages, and the development of computable representations is challenging.

This dissertation proposes and demonstrates a new representation model and an accompanying modeling methodology for representing building codes in computable form that can be utilized in the development of automated compliance checking systems. The model adopts the four level representation paradigm as a theoretical base and uses the semantic modeling approach for developing the building code representation. The model breaks down the representation into four levels which allows separate modeling of domain concepts, individual rule statements, relationships between rules, and the organization of the building code.

The dissertation 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. The applicability of the model has been evaluated through a case study. The case study successfully illustrates the modeling of building codes that constitute parts of İzmir Municipality Housing and Zoning Code, as well as a prototype implementation of an automated checking system utilizing this building code representation.

ÖZET

UYUMLULUK DENETİMİ OTOMASYONU İÇİN YAPI YÖNETMELİKLERİNİN BİLGİSAYARDA MODELLENMESİ

Yapı projelerinin ilgili yönetmeliklere uyumluluk denetiminin otomasyonu araştırma alanında geliştirilen bu doktora tezinde, yapı yönetmeliklerinin bilgisayar ortamında modellenmesi ve otomatik denetleme sistemlerinde uygulanmasına odaklanılmıştır. Uyumluluk denetimi sürecinin otomasyonuna yönelik sistem geliştirme çalışmaları, yapı projelerinin ilgili yönetmeliklere göre yetkili kurumlarca, zaman ve maliyet etkin olarak hatasız bir şekilde denetlenmesini hedefleyen teorik ve uygulamalı araştırmaların odağında yer almaktadır. Otomatik denetleme sistemlerinin geliştirilmesi için öncelikle yönetmeliklerin sayısal modellerine ihtiyaç duyulmaktadır. Yapılaşmaya ilişkin yönetmeliklerin birbiri ile ilişkili karmaşık kurallardan oluşan, sadelik ve düzenden uzak yapısından dolayı bu yönetmeliklerin bilgisayar ortamında sayısal olarak modellenmesi oldukça zorlu bir araştırma alanı olarak görülmektedir.

Bu tez çalışmasında, uyumluluk denetimi otomasyonuna yönelik sayısal yönetmelik modellerinin oluşturulması için yeni bir temsil modeli ve ona eşlik eden bir modelleme metodolojisi önerilmektedir. Önerilen model, düz yazı biçimindeki yönetmeliklerin kurgusal yapısının tanımlanmasına yönelik teorik çalışmaların sonucu olarak ortaya çıkan ve literatürde dört katmanlı modelleme paradigması olarak yer alan yaklaşıma dayanmaktadır. Bilgi modelleme yöntemi olarak ise son zamanlarda ortaya çıkan semantik modelleme yaklaşımı kullanılmaktadır. Önerilen model ile yönetmelikleri oluşturan terimlerin, kural cümlelerinin, kurallar arası ilişkilerin ve organizasyonel yapının ayrı katmanlarda modellenmesi sağlanarak sayısal yönetmelik modelleri için sistematik bir yöntem ortaya konmaktadır.

Bu tez çalışması, bir yapı yönetmeliğinin dört katmanlı olarak semantik tabanlı modellenmesinin uyumluluk denetimi otomasyonuna yönelik denetleme sistemleri için etkili bir modelleme stratejisi olduğunu göstermektedir. Önerilen modelin yapı tasarımı alanında uygulanabilirliği, İzmir İli Tip İmar Yönetmeliği örnekleminde gösterilmiştir. Yönetmeliğin yapı tasarımı ile ilgili bölümlerinin önerilen model çerçevesinde sayısal modeli oluşturularak bu model ile çalışan prototip bir otomatik denetleme sistem uygulaması geliştirilmiştir. Geliştirilen sistem çeşitli yapı projesi örnekleri üzerinde test edilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1. Problem Statement

In the Architecture, Engineering, and Construction (AEC) industry, every building project must be checked against numerous building codes for its compliance and is allowed to be executed only when the compliance with all applicable rules of the building code have been achieved. Compliance checking is a major task for both architects and building certifiers often involving ambiguities and inconsistencies in assessment, leading to delays in the overall construction process. Moreover, failure to correctly assess projects for compliance can have negative effects on building performance and allow errors that are expensive to fix.

Automated compliance checking has long 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 into developing automated compliance checking systems has focused mostly on representation of building codes in computational format, definition of building model views, compliance checking algorithms and reporting.

Automated compliance checking systems are expected to retrieve a set of building codes from related authorities and conduct compliance checking on submitted building projects. 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.

The main objective of this dissertation is to develop a building code representation, along with a formal methodology for modeling building codes based on this representation. A detailed analysis of The Housing and Zoning Code of the city of

Izmir in Turkey is presented as a case study. A computer representation of this code is built following the developed modeling methodology and is accompanied by a rudimentary automated checking system as proof of concept.

1.2. Motivation

This dissertation is motivated by the need to develop representations for building codes, based on which software tools could be developed. Having building code representations in a computable format may have the following benefits:

- Providing a basis for the development of automated compliance checking systems that can perform the checking of a building project more accurately, consistently, and rapidly.
- Providing help to building code authors in developing and updating building codes with a model that ensures a consistent set of rules without redundancies and contradictions.
- Providing assistance to architects in locating and accessing relevant rules among applicable building codes.

Because of these benefits, formally representing building codes in a computable format has been a highly active research area. Initial work was undertaken by Fenves (1966) on decision tables for representing building codes in a precise and unambiguous form. A follow-up project by the same group of researchers investigated the restructuring of the AISC (American Institute of Steel Construction) Specification (Nyman et al., 1973). Alternatives for restructuring the specification were presented, but the study concluded that appropriate restructuring, without changing the content, was not guaranteed (Nyman & Fenves, 1975). Subsequent studies investigated the potential of assisting building code authors in writing design standards (Fenves, 1976; Fenves & Wright, 1977; Harris & Wright, 1980; Rasdorf & Fenves, 1980). Later works focused on structuring building codes in a predicate logic structure (Jain et al., 1989; Rasdorf & Lakmazaheri, 1990). These research activities culminated in a software system, called SASE (Standard Analysis, Synthesis and Expression). This system aimed to provide tools for creating and checking decision tables, information networks, classification systems, and organizations within building code documents (Fenves et al., 1987).

Garrett and Hakim (1992) developed an object-oriented model of building codes, which allows organizing a building code around building objects pertinent to the building code. Waard (1992) offered another object-oriented approach to building code processing. In this study, an object model for residential buildings and another object model for building codes were developed, and the two models were linked for compliance checking. Yabuki and Law (1993) combined first order predicate logic and object-oriented modeling approaches to represent and process building codes. Kiliccote and Garrett (1998) developed a context-oriented model for representing building codes. This model uses the object-oriented modeling approach and organizes building code around "contexts" which are a collection of sub-classes used to define conditional parts of the provisions for which they are applicable. Given a predicate logic structure, Kerrigan and Law (2003) developed the REGNET application to determine the applicability of various codes under given building conditions, based on a question-and-answer user interface. These early efforts focused on the logical structure of building codes and how to represent them.

Exploration of building code compliance checking systems for building models began following the development of the Industry Foundation Classes (IFC) in the 1990s. Han et al. (1998) and Vassileva (2000) laid out general opportunities and a client-server approach. Han et al. (2002) later developed a simulation approach for disabled access checking. These efforts set the research field for larger, more industrial-based efforts (Eastman et al., 2009). The Singapore Construction and Real Estate NETwork (CORENET) project is the earliest production of building code compliance checking effort initiated in 1995 (Liebich et al., 2002). Initial work was based on electronic 2D drawings, but later on IFC (buildingSMART, 2008a) was used. The CORENET project developed the FORNAX platform to capture needed building code information. A more recent effort is the DesignCheck system from Australia, initiated in 2006 (Ding et al., 2006). In this effort, the EXPRESS Data Manager (EDM) (Jotne, 1994) platform was used for encoding barrier-free accessibility rules. Another effort, led by USA International Code Council (ICC), developed SmartCODES (Conover, 2007; Nisbet et al., 2009). It is a new platform providing methods of translation from written, natural language rules to computer code. The platform targets energy conservation rules. USA General Services Administration (GSA) have supported development of a rule checking system for circulation and security validation of U.S. Court houses (Eastman et al., 2008).

Although several researches have already proposed various building code models, and software environments to operate on these models and check building projects for compliance with building codes, for a variety of reasons, employing these environments AEC industry practice and general acceptance of them has been limited. A literature survey reveals that certain reasons for this failure are related to the building code models used in these environments and not necessarily 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.

The motivation for this dissertation stems from these limitations of previous building code models. In this regard, developing a more general (comprehensive, flexible, maintainable, and consistent) representation for building codes is highly desirable.

1.3. Objective

The goal of this dissertation is to develop a new computer representation for building codes that can be used in the development of automated compliance checking systems. The aim is to develop a formal model that supports the creation of digital representations of build codes by following a corresponding methodology.

A new model for representing building codes is needed to improve on existing models. After investigating previous models and identifying their limitations, the following requirements were established for the new representation model:

Independence - Keeping the representation of building codes independent of the compliance checking system and the design system. The model should not be hard coded into the systems to avoid the modification of the whole system when the building code is changed. The model should be portable between automated checking systems and code authorities.

Conciseness - Avoiding redundancy in the representation of building codes. The same object should not need to be defined more than once in the model.

Consistency - Preventing ambiguities as well as contradictions among rules. Building codes are complex and complicated documents which include rules that are open to interpretation, uncertain, sometimes even contradictory. The model should ensure consistency.

Comprehensiveness - Representing all of the various types of information in building codes (i.e. concepts, requirements, applicability conditions, etc.).

Maintainability - Allowing creation and addition of new rules and modification of existing ones. 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.

The new representation model aims to meet the above requirements. It is intended to be utilized as a basis for the development of automated compliance checking systems. In addition, the model can also enhance and facilitate building code authoring. To utilize the representation model and create digital versions of existing building codes, a building code modeling methodology is proposed. The feasibility, benefits and limitations of the developed representation model should be validated with a realistic case study.

1.4. Methodology

There is a long history of interest in the study of building code representations. It started with initial efforts to organize the logical structure of building codes in the late 1960's. Efforts on the automated application of building codes to digital building representations came in the 1990's (Eastman et al., 2009). Research efforts so far, have emphasized representation models and prototype implementations. The discussions on methodological issues have centered on reasons for selecting particular information modeling techniques, and not so much on the method of defining representation models.

The literature survey conducted as part of this dissertation (see sections 2.2 and 2.3) revealed that the research strategy that de facto has been followed in most research efforts consisted of the stages listed below:

- Selecting a particular information modeling methodology or language to be used for constructing the representation models.
- Analyzing the characteristics of building codes.
- Definition of data structures for representation models.
- Experiments with prototypes to test the representation models.

The methodology of this dissertation adopts these research stages used by the majority of previous researchers. This dissertation has been conducted in the following stages:

- Exploring and evaluating previous modeling approaches.
- Analyzing building codes to understand the various types of information contained in them; identifying the components of rule statements as well as the organization of the documents.
- Developing a formal representation model for building codes.
- Defining a building code modeling methodology for utilizing the representation model to build digital versions of existing building codes.
- Modeling of an actual building code as a case study to demonstrate the feasibility, benefits, and limitations of the representation model.
- Implementing a prototype system to demonstrate an application and to test the validity of the new model.

As a conclusion, this research proposes a formal model for building code representation based on the analysis of building codes and theories established in literature. The theory embedded in this proposed model is evaluated through the development of an actual building code and a prototype implementation. The study is mostly concerned with the modeling process for building code representations even though the impact of utilizing various information modeling techniques (such as object-oriented modeling, ontologies, semantic modeling) have been considered.

1.5. Scope

The foreseen advantages of automated systems for compliance checking of building projects against applicable building codes over the traditional, manual compliance checking processes has been well established in literature (Fenves & Wright, 1977; Han et al., 1997; Eastman et al., 2009; Hjelseth, 2009; Zhang & El-Gohary, 2011). These advantages form the main motivation behind pursuing a working solution for automated systems, yet it is clear that such a system is still elusive. The main goal of this research was not to attempt the development of an automated compliance checking system but to identify the next steps to be taken.

Following a careful investigation of literature, shortcomings in representing building codes were identified. IMHZCode was chosen as a code document since it is representative of complex building codes in effect. The analysis of IMHZCode led to the classification of rules according to their formalizability and self-containedness. In developing a new model that addresses the shortcomings identified in previous efforts, in this study formalizable rules were considered. Non-formalizable and semi-formalizable rules (which made up 20% of the rules) were left out of scope. The dissertation focuses on developing a modeling methodology for formalizable rules as a first step before attempting to automate reasoning with non-formalizable and semi-formalizable rules that carry ambiguities.

The study was also restricted to analyzing and modeling the section of IMHZCode containing rules that apply to all types of buildings. Rules related to permits, the construction process and specialized buildings have not been included in the case study. Furthermore, only the IMHZCode was modeled and automated checking against multiple codes from different domains was not considered.

While developing an automated checking system was not within the scope of the study, an experimental prototype was developed, successfully demonstrating the feasibility of the new building code representation model. The demonstrative examples cover testing of simple rules (e.g. doors) as well as rules with complex conditions (e.g. buildings, setbacks).

1.6. Outline

The thesis is organized into five chapters. This chapter provides a brief introduction of the thesis. It covers the problem statement, research motivation, objectives, methodology, and scope of this dissertation.

CHAPTER 2 contains the background information for this research. It describes analytical knowledge about the building codes and the related literature review. The review covers previous and current works related to this dissertation. Several models and approaches for building code representation and compliance checking systems are examined, evaluated, and discussed. CHAPTER 2 also introduces current technologies for implementation of building code models, and assesses their advantages and limitations.

CHAPTER 3 describes in detail the developed representation model and modeling methodology for creating digital building codes. CHAPTER 4 addresses the evaluation of the developed model by a detailed discussion on the case study illustrating a building code model and the prototype system implementation performed for the developed model.

CHAPTER 5 concludes the dissertation by providing a summary, conclusions, and possible future research directions.

CHAPTER 2

BACKGROUND

This thesis constitutes a study in the field of automated building code compliance checking, with a concentration in the topic of building code modeling. Development of automated building code compliance checking systems primarily requires appropriate building code models. However, the process of modeling building codes as computable rule sets is not straightforward due to the complex structure of building codes. Building codes include rules that are open to interpretation, uncertain, sometimes even contradictory and hence impossible for modeling. For all these reasons, it is a challenge to propose a method for representing building codes in computable format for automated compliance checking. Before proposing a method for developing building code models, understanding the nature of building codes and reviewing previous research on modeling building codes and implementing automated compliance checking systems based on these models is important.

This chapter introduces the related background information for this thesis. It starts with a general introduction to the study on building codes. Then, section 2.2 presents an overview of previous building code models and section 2.3 presents the relevant automated code compliance checking implementations and gives a summary of automated code compliance checking process and existing platforms supporting these systems.

2.1. Building Codes

In this thesis, the term "building code" is used to refer to any formal document for the evaluation of building projects. Formal documents include building codes, regulations, standards, and specifications. A building code is generally considered as 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.

2.1.1. Types of Building Codes

Building codes are determined by appropriate authorities in different domains and may vary widely from country to country. The practice of developing, approving, and enforcing building codes varies considerably among nations. Many countries have national building codes, developed by government agencies and applied to all building and construction work across the country. There are instances when some local jurisdictions choose to develop their own building codes.

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)

In addition, individual municipalities have their own housing and zoning codes that include the rules defined by the ministry documents and add further specifications.

2.1.2. Elements of Building Codes

Building codes are complex written documents. 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. They are composed of hierarchically organized parts, chapters, and clauses that contain a number of statements. In general, building codes includes following three types of clauses:

- General provisions
- Definitions
- Prescriptions

General provisions include statements that 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:

> "Clause 2 – Scope:
> This code, prepared in accordance with Construction 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."
>
> "Madde 2 – Kapsam:
> 3194 sayılı İmar Kanunu ve Büyükşehir Belediyelerinin Yönetimi Hakkında 3030 Sayılı Kanunun Uygulanması İle İlgili Yönetmeliğin 8.maddesi gereği hazırlanan bu yönetmelik, İZMİR BÜYÜKŞEHİR BELEDİYESİ belediye ve mücavir alan sınırları içinde uygulanır."

Definitions include statements that explain clearly the specific names used in the building code and give detailed descriptions about terms. For example, the following expression quoted from IMHZCode defines meaning of a name (High-rise building):

> "Clause 18 – Definitions Related to Buildings:
> d – High-rise building: A building, height of which is greater than 30.80 meters or has more than thirteen (13) storeys."
>
> "Madde 18 – Yapıya İlişkin Tanımlar:
> d – Yüksek Yapı: Son kat tavan döşeme kotu 30.80 metreyi ve/veya bodrum kat dahil olmak üzere toplam kat adedi 13'ü aşan (13 kat hariç) yapılar Yüksek Yapı olarak kabul edilir."

Prescriptions include statements that define constraints about physical building components, spaces and relations. For example, the following expression quoted from IMHZCode defines a constraint about side setbacks:

“Clause 27 – Distance of Setbacks

B - Side Setbacks: Side setbacks (up to and including 4 storeys) shall be 3.00 meters. For buildings taller than this side setbacks are increased by 0.5 meters for every additional storey. However, for timber-framed buildings side setbacks must be at least 5.00 meters. “

“Madde 27 – Bahçe Mesafeleri:

B- Yan Bahçe Mesafeleri : Yan bahçe mesafeleri (4 kata kadar 4 kat dahil) 3.00 m. olacaktır. Bundan yüksek katlı binalarda yan komşu mesafeleri 3 m.ye beher kat için 0.50 m. ilave edilmek suretiyle tespit olunur. Ancak, ahşap karkas yapılar için en az 5 m. mesafe bırakılması şarttır.”

Prescriptions are the only types of clauses that are applied to submitted building projects while conducting compliance checking. From this point, a clause denotes a prescription. Clauses are composed of a number of statements. While some of these statements relate to clarifications about calculation methods or applicability conditions, others relate to rules indicating quality requirements that must be satisfied by a building project. Automated compliance checking systems applies the rules to a proposed project to evaluate the compliance.

Building codes are written text documents, to be interpreted by humans. They are not necessarily structured in a strict and straightforward manner that can be interpreted by machines. They have complex structures. While some simple rules can easily be defined in a single statement, others require a series of statements making exceptions, clarifications and modifications. 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. Since automated compliance checking requires a rule-based system, which applies rules to a proposed project to evaluate the compliance with all applicable rules of the building code, it is important to document how much of the building code and which rules can be modeled reliably in automated compliance checking systems.

2.2. Building Code Models

There has been an extensive amount of research conducted internationally over the last four decades in the area of representing building codes in computable format for automated compliance checking. This section presents previous models for building code representation and evaluation that have been developed, explores common themes and different approaches used, and compares the strengths and limitations of the major building code models. The models presented here are:

- Decision tables and SASE model
- Rule-based models
- Logic-based models
- Object-oriented models
- Hybrid models
    - Object-logic model
    - Context-oriented model
- Semantic models
- Ontology-based models

Each of these models is explained in the next six sections.

2.2.1. Decision Tables and SASE Model

The introduction of decision tables by Fenves (1966) is the initial effort on building code modeling. In this effort, building code provisions are represented in a precise and unambiguous decision table form. A decision table is a concise tabular representation of the conditions applicable in a given situation and of the appropriate actions to be taken as a result of the values of the conditions (Fenves et al., 1969). Decision tables can explicitly represent individual provisions as a set of conditions on data items. A data item is a precise identification of an information element occurring in a building code (e.g. height of a building). A decision table is divided into four sections as shown in Figure 2.1.

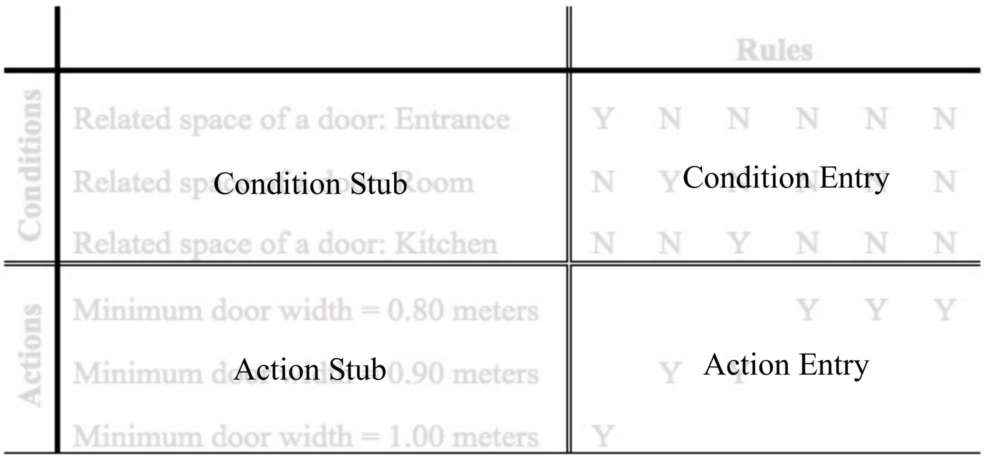


Figure 2.1. Sections of a decision table

The *condition stub* section in the upper left is a list of the conditions that have a bearing on the outcome. The *condition entry* in the upper right-hand section of the table lists the pertinent combinations of the conditions in columns. Each column specifies a rule of a provision. The lower left section of the decision table is the *action stub,* listing all the possible actions that may be taken. The lower right-hand section of the table is the *action entry*, specifying the particular action or actions to be taken corresponding to the specified rule. The elements of the *condition entry* can have only one of three possible values, i.e., "Y" indicates the condition is true, "N" indicates the condition is false, "I" indicates the condition is immaterial. The elements of the action entry may be either "Y" means that the corresponding action is to be executed, or "blank" means that the action is not to be executed (Fenves et al., 1969).

The provisions of building codes can be represented by means of decision tables. How to use decision tables to model building code provisions can be illustrated using the following example of a provision for computing the minimum width of a door based on the IMHZCode, shown in Figure 2.2. A decision table for determining the data item "minimum door width" in the provision is shown in Figure 2.3. In this example, the upper section of the decision table (condition stub and condition entry) represents conditions, i.e. various usage types of related space of the door, while the lower section (action stub and action entry) represents the actions corresponding to various conditions. For example, according to the last column in the decision table, if a door's related space in a given building project is a bathroom, the door width should not be less than 0.80 meter.

"Clause 47 – Doors:

B – Door Width: [...] Clear width of entrance doors of independent unit shall be at least 1.00 meters. Clear width of room and kitchen doors shall be at least 0.90 meters. Clear width of bathroom, WC, cellar doors shall be at least 0.80 meters. […]"

"Madde 47 – Kapılar:

B- Kapı Genişlikleri : [...] Bağımsız bölüm kapıları, kasa dahil (1.00) metreden, Oda ve mutfak kapıları kasa dahil (0.90) metreden, Yıkanma yeri, WC, odunluk, kömürlük, kiler kapıları kasa dahil (0.80) metreden az olamaz. [...]"

Figure 2.2. Clause 47.B from the IMHZCode

		Rules					
Conditions	Related space of a door: Entrance	Y	N	N	N	N	N
	Related space of a door: Room	N	Y	N	N	N	N
	Related space of a door: Kitchen	N	N	Y	N	N	N
	Related space of a door: Cellar	N	N	N	Y	N	N
	Related space of a door: WC	N	N	N	N	Y	N
	Related space of a door: Bathroom	N	N	N	N	N	Y
Actions	Minimum door width = 0.80 meters				Y	Y	Y
	Minimum door width = 0.90 meters		Y	Y			
	Minimum door width = 1.00 meters	Y					

Figure 2.3. An example of using a decision table to represent building codes

Decision tables can concisely represent all the possible combinations of conditions and related actions of each provision. However, decision tables do not address the overall organization (including relationships among provisions) of a building code. Fenves et al. (1987) addressed this deficiency with the development of a software system called SASE (Standards Analysis, Synthesis and Evaluation) to provide tools for creating and checking more comprehensive building code models. This software is based on a four-level representation methodology to represent individual provisions, relationships among provisions, and the organization of the building code. This methodology is derived on the basis of an abstract model of the logical structure of building codes which is developed by Nyman et al. (1973) who investigated possible methods of restructuring building codes. According to this methodology, the content of the building code is examined in following four levels:

- The top level (organizational network) that represents outlines and organization of the building code.
- The intermediate level (information network) that represents the dependency relationship among provisions.
- The detailed level that represents the individual provisions in the form of decision tables.
- The lowest level that consists of the basic *data items* referred to in the provisions.

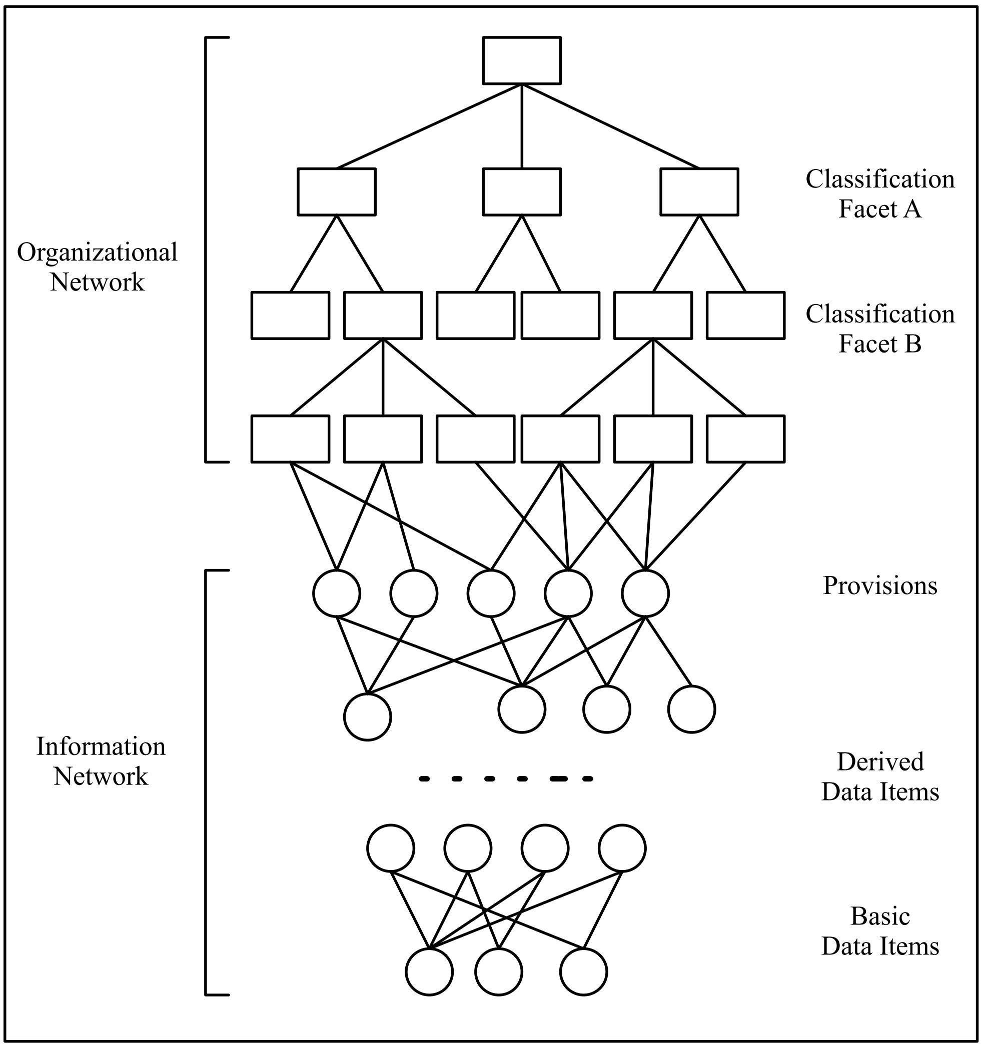


Figure 2.4. The SASE model
(Source: Fenves et al., 1987)

Figure 2.4 shows the skeletal architecture of the SASE model (Fenves et al., 1987). In this model, each information element occurring in a building code is identified by data items. Data items represent all the variables in the building code. A data item may be one of following four types:

- A numeric quantity such as "H" (building height).
- A specific value of "valid", "invalid", or "not applicable".
- A Boolean value of "Yes" or "No".
- An enumerated value such as "attached", "semi-detached", or "detached".

A data item is classified into two: a basic data item, and a derived data item. While a basic data item has no ingredients from within the building code to determine its value, a derived data item has both ingredients and dependents to derive its value.

In the SASE model, each individual provision within the building code is represented by decision tables as a set of conditions on data items. Each decision table is responsible for producing a value for only one data item. The decision tables used in the SASE model are restricted to limited entry decision tables in which condition values are restricted to Y, N, or I.

In the SASE model, an information network is used to represent the precedence relationships among the data items of the building code. The network is composed of nodes and branches. Each node represents one data item in the building code. Each branch represents a relationship between two nodes (data items). The data item on top of a branch is commonly referred to as parent and the lower level one is referred to as a child. A data item may have more than one child and more than one parent. If a data item does not have a child, it is a basic data item. If a data item has at least one child, it is a derived data item. For each information network, there should be exactly one derived data item that has no parent which is called the terminal data item of the network. In any case, all of the basic data items must be present before derived data items can be evaluated. This is called the precedence relationship that must be observed in an information network. As an example, an information network of the provision for a door, as described in Figure 2.2, is shown in Figure 2.5. In this figure, the node farthest to left, which does not have any successor nodes (parents), such as "Provision for door width" is called a terminal data item or a provision. The intermediate node such as "relSpaceUsage" represents derived data items. The nodes farthest to the right, which do not have any predecessor nodes (childs), such as “bathroom, “room”, ‘or “kitchen” are basic data items.

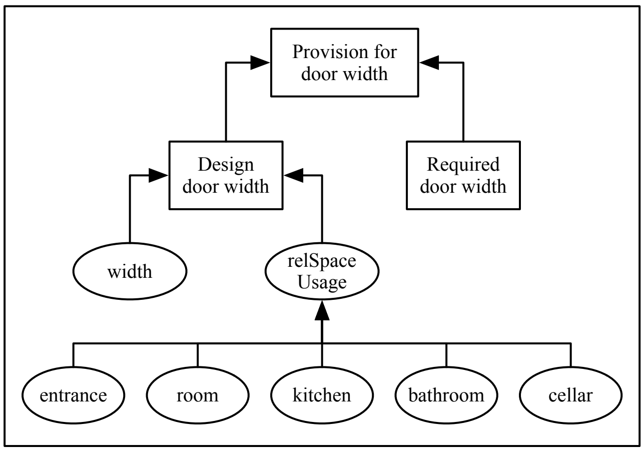


Figure 2.5. Information network for the provision for door width

In the SASE model, at the top-level, the building code is organized in a systematic manner such that individual provisions can be accessed easily. The classification tree is used to classify top-level data items of the building code, i.e., the provisions. Each provision is associated with one or more leaves of the classification tree. The methodology for classification of provisions is based on the faceted classification system consisting of several independent areas such as fields and facets. A field is a subject area (e.g., as physical elements) and a facet is a way to classify within a particular field (e.g., door, window). As an example, partial facets from two fields, “physical elements” and “dimension constraints”, are shown in Figure 2.6 and two examples of possible outlines for the code are shown in Figure 2.7.

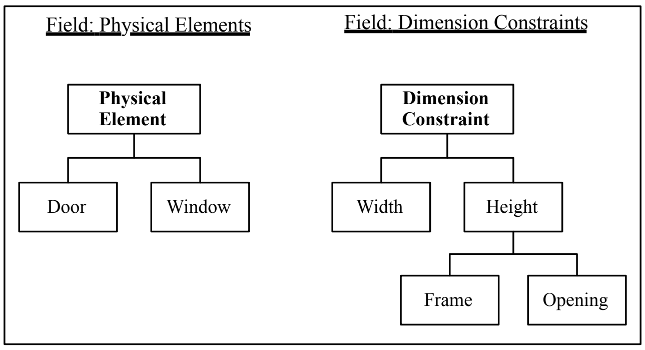


Figure 2.6. Partial facets of two fields
(Source: Adapted from Garrett & Fenves, 1987)

Possible Outline A:

```
Ch. Section   Subsection          Prov.
1. Door
     1.1. Width -------------------- Pr.1:...
     1.2. Height -------------------
               1.2.1. Frame ------ Pr.2:...
               1.2.2. Opening ---  Pr.3:...
2. Window
     2.1. Width -------------------- Pr.4:...
     2.2. Height -------------------
               1.2.1. Frame ------ Pr.5:...
               1.2.2. Opening ---  Pr.6:...
```

Possible Outline B:

```
Ch. Section   Subsection          Prov.
1. Width
     1.1. Door --------------------- Pr.1:...
     1.2. Window ----------------- Pr.4:...
2. Height
     2.1 Frame
               2.1.1. Door -------  Pr.2:...
               2.1.2. Window ---  Pr.5:...
     2.2 Opening
               2.1.1. Door -------  Pr.3:...
               2.1.2. Window ---  Pr.6:...
```

Figure 2.7. Two possible organizational outlines for the building code

SASE Model's main contribution has been to introduce a code model that was independent of the conformance checking system. This has allowed non-programmers to modify the code without updating the processing system. Decision tables are simple tools to provide a code description to the system. Furthermore, the SASE Model also provides dynamic reclassification of the code allowing different users to examine the code according to their needs.

However, the main problem with the SASE Model is that it uses data items as the primary element of building codes and that it lacks definition of building objects to which the building codes apply. This leads to a high number of data item definitions and complexity of the relationships among data items increase quickly as the number of provisions increase. Moreover, decision tables require complicated data item definitions. A data item needs to define both the reasoning methods and conditions associated with the specific building object that the data item is describing.

2.2.2. Rule-Based Models

Several researchers (Rosenman & Gero, 1985; Dym et al., 1988; Rasdorf & Wang, 1988; Kumar, 1995) have proposed methods based on a rule-based modeling approach for representing building codes as rules/clauses in processing systems. In these models, the clauses of the building code are represented as a set of rules in the form of IF [condition] THEN [action] statements instead of decision tables. The IF statements of the rules describe applicability conditions which need to be satisfied. The THEN statements describe required actions to be taken. This approach is a more natural way for building code representation than decision tables. Figure 2.8 shows the example of using rule-based approach to represent a provision of IMHZCode for doors given in Figure 2.2.

```
Rule 1:
        IF Related space of a door is an entrance
        THEN Check the width of the door is equal to 1.00 meters.
Rule 2:
        IF Related space of a door is a room or kitchen
        THEN Check the width of the door is equal to 0.90 meters.
Rule 3:
        IF Related space of a door is a cellar, WC, or bathroom
        THEN Check the width of the door is equal to 0.80 meters.
```

Figure 2.8. An example of using rule-based approach to represent building codes

Rule-based representations are more expressive than other representational approaches. The main advantage of this approach representing building codes as rules is ease of implementation in the building code processing systems (Rasdorf & Wang, 1988). Moreover, rule representations match the actual building code rules so that checking the completeness of the model is trivial (Rosenman & Gero, 1985).

However, modeling a building code by using the rule-based approach is likely to end up with a large number of rules that could be unmanageable in practice because one provision needs to be represented by more than one rule. The main handicap in this approach is that the rules (IF and THEN parts) have to be modeled in specific programming or modeling languages into a building code processing system. This is a severe disadvantage for the approach since its ability to accommodate changes becomes limited (Rasdorf & Wang, 1988). The rules that represent the building code clauses are "hard-coded" into the system which means that they are not separated from the programming codes of the building code processing system. This hard-coding of the rules into the system reduces the efficiency of system maintenance when the rules need to be updated due to the changes in the building code (adding new clauses, and changing or deleting existing clauses). Such revisions of the system may be difficult and expensive, especially with programs of considerable size. Additionally, since the systems are not understandable and accessible by users who do not have any programming knowledge the system is the dependent on the system programmer for downstream modifications.

While the limited ability to accommodate changes is one deficiency, there is also the difficulty for code experts to ascertain the correctness of the program code. In most cases, the programmer of the building code processing system is not the building code author, thus there is a chance of misinterpretation. The building code processing system is prone to misinterpretation errors in which case the system will not perform its prescribed compliance checking function correctly.

2.2.3. Logic-Based Models

Some researchers have investigated the use of predicate logic to model building codes. Jain et al. (1989) proposed a logic-based model, which is based on the SASE model, using predicates to represent the building code provisions instead of decision tables. This model provides validation and verification methods for checking completeness and uniqueness (lack of redundancy and lack of contradiction) of a building code model. In this project, limited-entry decision tables are converted into predicate logic statements. Building code clauses are represented as groups of statements of the form R_i: $C_i \rightarrow A_i$, where C_i represents the part to the left of the implication (condition) while A_i represents the part to the right of implication (action). Each group represents rules for a single data item.

Rasdorf and Lakmazaheri (1990) also proposed a logic-based approach for representing and processing a building code that extends the utility of the SASE organizational model for conformance checking. In this work, the overall organization of a building code is formally modeled using predicate logic. Predicate logic is used for developing a formal language for representing the organizational model of the building code. Using the formal language, a set of axioms (statements) is developed that represents the relationships between the classifiers and the clauses of the building code model. The formal language and axioms constitute the formal organizational model of the building code. Processing the organization of the building code is accomplished using the formal language to first formulate expressions called theorems and then to prove the theorems via what is called "the resolution theorem-proving strategy". Theorems are used to prove the uniqueness -meaning that the code model generates one and only one result when applied in any one situation- and completeness -meaning that the code model can be applied to all possible situations within its scope- of the building code (Rasdorf & Lakmazaheri, 1990).

Logic-based approaches adopt predicate logic to model the provisions of building codes. Predicate logic provides a formal, well-known and systematic knowledge representation language to express statements in well-formed formulas. In logic-based models, IF-THEN logic is also used to represent the building code clauses. The difference from the rule-based models is that the logic-based models use predicate logic. As an example, the example of IF-THEN rule shown in Figure 2.8 can be represented in predicate logic as shown in Figure 2.9. In this example, the applicability conditions is determined on the left side of the arrow. The action statement is specified on the right side of the arrow.

R_1: $\forall x(\text{Door}(x) \wedge (\text{RelatedSpaceUsage}(x)=\text{Entrance})) \rightarrow \exists(x)(\text{Width}(x) \geq 1)$

R_2: $\forall x(\text{Door}(x) \wedge (\text{RelatedSpaceUsage}(x)=\text{Room} \vee \text{Kitchen})) \rightarrow \exists(x)(\text{Width}(x) \geq 0.90)$

R_3: $\forall x(\text{Door}(x) \wedge (\text{RelatedSpaceUsage}(x)=\text{Cellar} \vee \text{WC} \vee \text{bathroom})) \rightarrow \exists(x)(\text{Width}(x) \geq 0.80)$

Figure 2.9. An example of using logic-based approach to represent building codes

Hakim and Garrett (1993) claimed that logic-based approaches enable the evaluation of the consistency, completeness, and clarity of building code models as well as support the reasoning about incomplete knowledge. However, logic-based approaches require building code modelers to have knowledge of logic to model building codes and understand them. Moreover, logic models include user-defined predicates and logical operators that prevent them from being widely implementable.

2.2.4. Object-Oriented Models

Hakim and Garrett (1992) pointed out one deficiency in previous building code models, which use data items as the primary data structure, concerning a lack of a formal model of the building objects within the scope of the building code. This leads to many data item definitions. Moreover, the definition for each data item must not only define the evaluation method, but also the conditions associated with the specific object of which data item is an attribute. This leads to complicated data item definitions. In both cases the models are hard to maintain and build. To address these issues, Garrett and Hakim (1992) developed an object-oriented model of building codes, which allows organizing a building code around building objects pertinent to the building code. This object-oriented model is composed of following four main groups of objects (see Figure 2.10):

- The design object hierarchy that represents the subclass relationships between objects of the building code. Objects define design-specific attributes and hierarchies between these attributes such as shape, function, material.
- The performance-limitation hierarchy that represents the limitation of the building code objects.
- The data item hierarchy that represents the types of information that are found in all building codes (e.g., a constant, a rule, or a table).
- The data item instance network which represents instances of the different types of information present with a specific building code.

Garrett and Hakim's incorporation of design objects within the analysis process provides a logical extension to the SASE methodology.

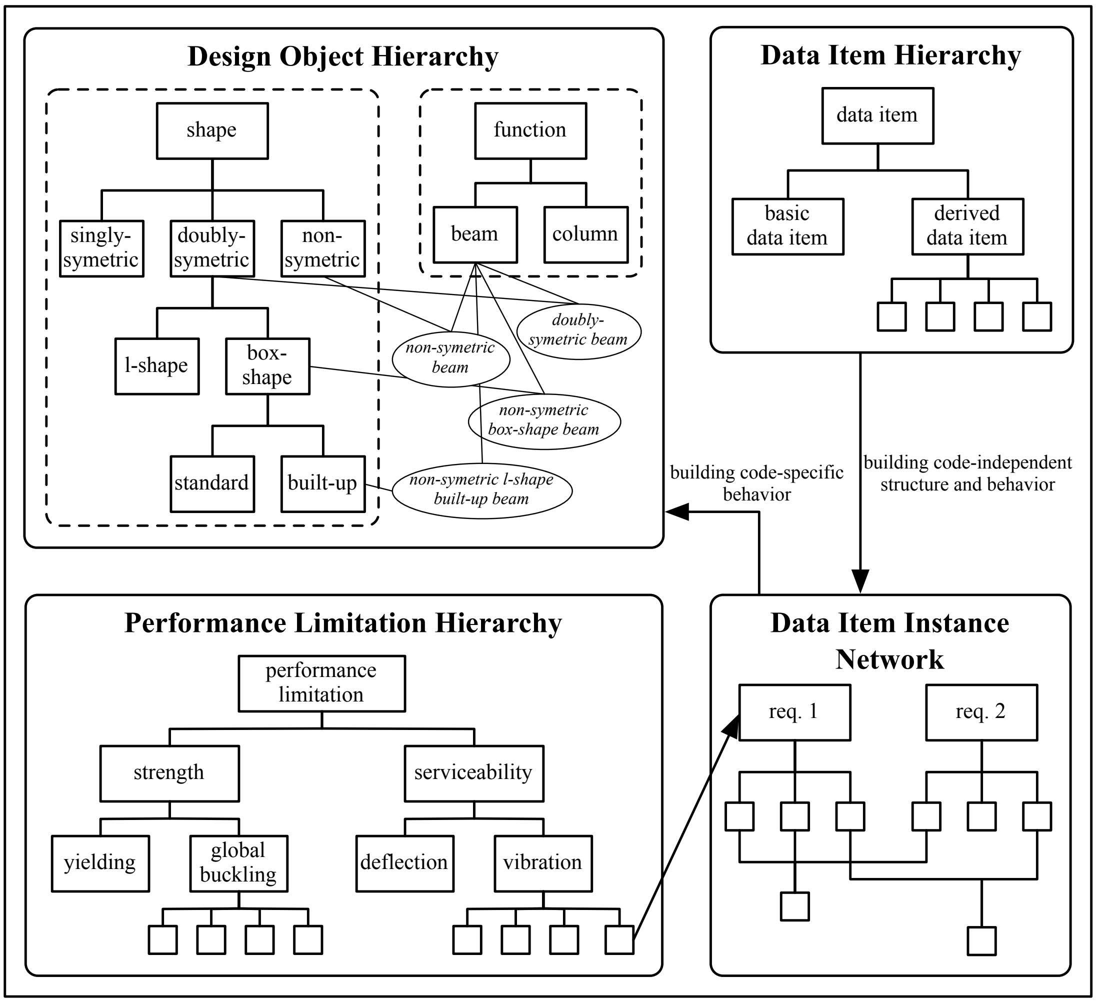


Figure 2.10. Object-oriented model of the building code
(Source: Hakim & Garrett, 1992)

The object-oriented modeling approach taken by Waard (1992) proposes the use of information models. This approach requires an information model of residential buildings, an information model of building codes, and a link between both information models. In this work, first, the information model for residential buildings containing architectural objects (e.g., rooms, walls, doors) is developed. Waard (1992) states that the information model can be also used as a neutral model for information exchange between participants in the building process, and between design teams and building authorities. After the construction of the residential building information model, the information model of building codes is developed. It is stated that the building code model should reflect the idea that building codes not only consist of constraints, but also of a model of a building according to those building regulations. To illustrate this point, De Waard modeled several provisions extending and adjusting the relationships developed in the residential building information model. At the last stage, the link between the building code information model and residential building information model is defined by describing the way in which building code views can be derived from the residential building information model. The model of the objects that are subject of the building codes form a building codes view on the residential building model.

Object-oriented models utilize class hierarchies to represent building codes, including their applicability conditions and requirements, so that the applicability of one section of a building code can be passed down to its subsections. Object-oriented models use classes and attributes to represent data items (concepts) of building codes, such as representing the data item “door” as a class and the data item “door width” as the class’s attribute. As an example of how such an object-oriented model can be developed, consider the clause in Figure 2.2, taken from IMHZCode, which specifies the minimum width of a door. Figure 2.11 shows the object oriented model constructed from the classes and their attributes mentioned in the clause, together with their relationships. Figure 2.12 shows the object-oriented interpretation for the clause.

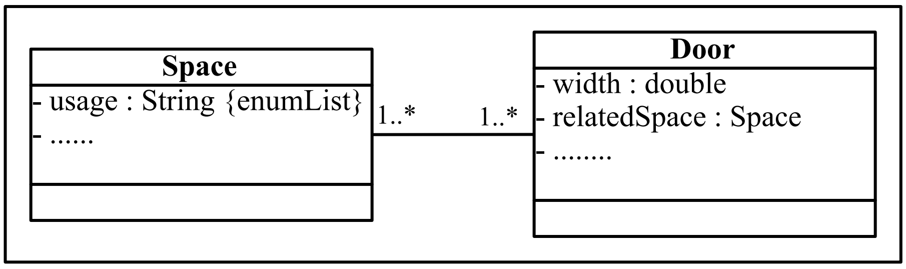


Figure 2.11. Object-oriented model from analysis of Figure 2.2

CLAUSE 47 – Doors
B – Door Width:

Classes:

- Door ()
- Space ()

Class Attributes:

- Door () {width, type, relatedSpaces[] }
- Space () {usage}

Relationships between Classes:

- relatedSpace (Door, Space)

Methods:

- setDoorType (Door) {

```
    IF Door.relatedSpace.usage equals to “entrance”
    THEN this.type = entranceDoor;

    IF Door.relatedSpace.usage equals to “room”
    THEN this.type = roomDoor;

    IF Door.relatedSpace.usage equals to “kitchen”
    THEN this.type = kitchenDoor;

    IF Door.relatedSpace.usage equals to “bathroom”
    THEN this.type = bathroomDoor;

    IF Door.relatedSpace.usage equals to “cellar”
    THEN this.type = cellarDoor;
}
```

- checkDoorWidth (Door) {

```
    IF Door.type equals to “entranceDoor” &
        Door.width ≥ 1.00
    THEN return “this door is valid”;

    IF Door.type equals to “roomDoor” || “kitchenDoor” &
        Door.width ≥ 0.90
    THEN return “this door is valid”;

    IF Door.type equals to “bathroomDoor” || “cellarDoor” &
        Door.width ≥ 0.80
    THEN return “this door is valid”;

    ELSE return “this door is not valid;
}
```

Figure 2.12. An example of an object-oriented interpretation of Figure 2.2

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.

2.2.5. Hybrid Models

Hybrid models combine some of the previous representation approaches. An example is the Object-Logic model (Yabuki & Law, 1993) which combined first order predicate logic and object-oriented modeling approaches to represent and process building codes. They claimed that object oriented approach is suitable for representing the organization and data items of the building code and the logic programming approach is suitable for representing and processing building code provisions. The model they proposed also allows the development of formal procedures to check completeness and uniqueness of the building code and conflicts among the provisions. The system framework for the object-logic model (Figure 2.13) consists of two submodels: object-logic model, and hyper document model. These two models are integrated by sharing methods objects representing the design provisions. The hyper document model contains the provisions, background information of building codes, external programs, and method objects. The hyper document model serves as a large document storage system for building code provisions. The object-logic model consist of the following 5 basic modules (Yabuki & Law, 1993):

- A standards base that consists of a class hierarchy representing the organization and provisions of the building code. Provisions are modeled as *method objects* which are a set of object-logic sentences.
- A CAD object database that holds member definitions and the object model and facilitates retrieving member attributes from an engineering database.
- A conformance checking module that performs compliance checking of a given member with applicable provisions.
- A component design module that generates component design of a given member.

- A standards analysis module that checks completeness and uniqueness of provisions. This module also performs simple analysis on the provisions by checking whether the relationships among the method objects exist.

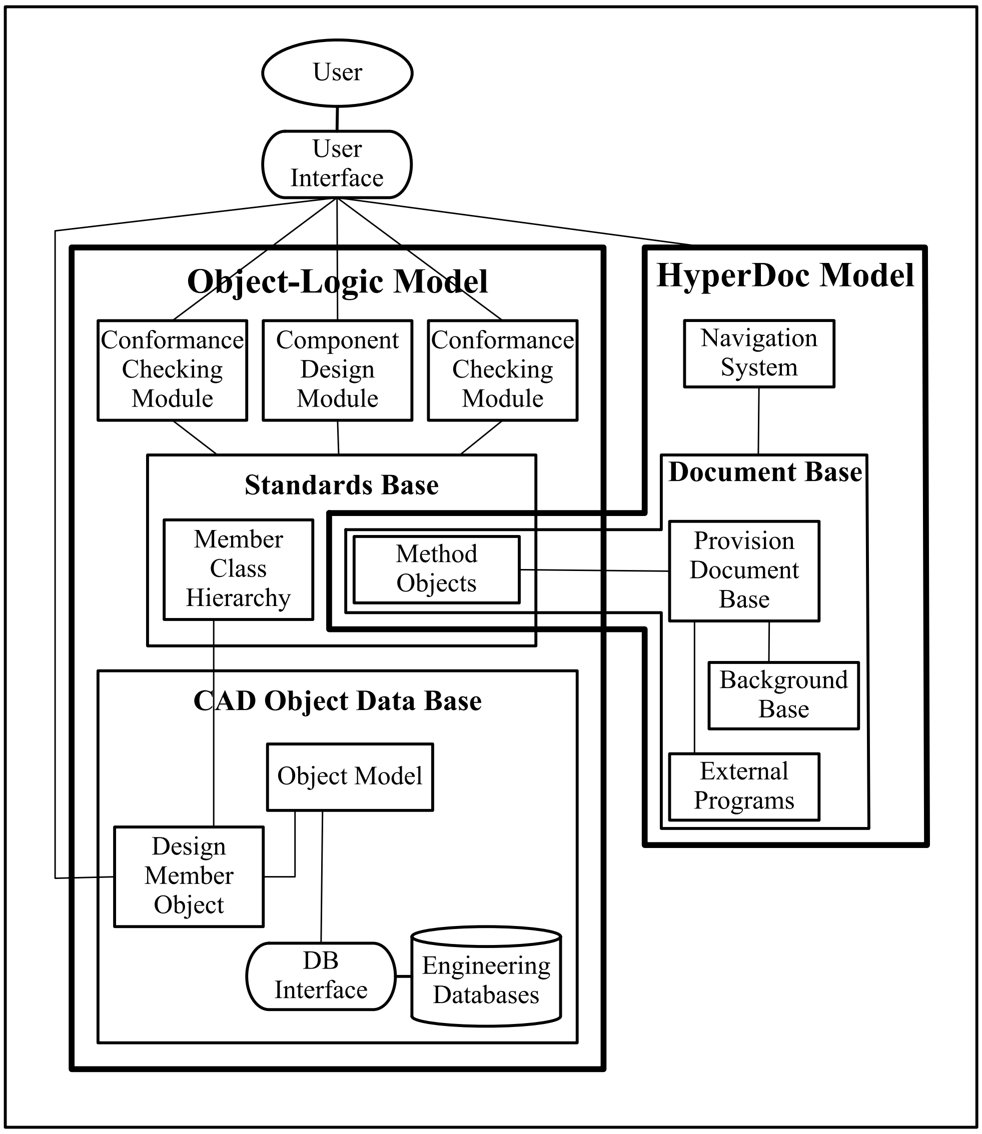


Figure 2.13. Overview of the object-logic model
(Source: Yabuki & Law, 1993)

Kiliccote et al. (1994) points out the object-logic model has similar complex classification hierarchy issues as the object-oriented models has which make working with building code cumbersome. In these models a class hierarchy is used as a method to organize the building code and provisions are associated with the classes in this hierarchy. These models require many classes and complex multi-parent subclasses which are created in the lower parts of the class hierarchy to represent the specific contexts to which the provisions in the building code apply (Kiliccote et al., 1994). As a result, both object-oriented and object-logic models become insufficient to represent applicability concepts in the building code provisions.

Kiliccote et al. (1994) addressed this issue with the development of a Context-Oriented model. This model also uses the object-oriented approach but addresses the problem of complexity of the object-oriented model's class hierarchies by organizing the building code around "contexts". A context is a collection of sub-classes in the classification hierarchies to which the provisions are associated. These contexts are used to define conditional parts of the provisions for which they are applicable. Based on the context-oriented modeling approach, (Kiliccote & Garrett, 1998) proposed a multi-module distributed framework, which is called the Standards Processing Framework (SPF), to represent and reason with building codes. The modules in the SPF are called SPF Agents using different representation and reasoning methods to develop general models. In the SPF, an agent is a self-contained entity that communicates with another through messages expressed in a common language. As part of this work, the SPF communication language (SPF-CL) is developed to allow interaction between various SPF-Agents. SPF-CL is used to describe and ask for:

- Information about agents (e.g., to define that there is an agent that can compute the maximum allowable height of a building).
- Building codes and individual provisions (e.g., to define requirements, including their applicability conditions, and data needs).
- Design information (e.g., to specify the total gross area of a building).

To meet these functions, SPF-CL is composed of three languages: the Agent Description Language (ADL), the Standards Modeling Language (SML), and the Standards Usage Language (SUL). ADL is used to describe knowledge about agents. SML is used to describe the content of a building code to an agent. SUL is used to describe the design to an agent and request information about the result of the evaluation of a building code provision against that design. SPF-CL uses the following four major objects to model building codes:

1. *Concepts* which are used to represent real world concepts (e.g., structure, building, room, door). In SPF-CL, concepts are defined using a define Concept command as shown in the next example.

```
define Concept {
        name = “Room”;
};
define Concept {
        name = “Door”;
};
```

2. *Classifications* which is a collection of distinct concepts that differ from each other by some set of properties. In SPF-CL, classifications are defined using a define Classification command as shown in the next example.

```
define Classification {
        name = “Space”;
};
define Concept {
        name = “Entrance”;
        is_a = Space;
}
define Concept {
        name = “Room”;
        is_a = Space;
}
define Concept {
        name = “Kitchen”;
        is_a = Space;
}
define Concept {
        name = “Bathroom”;
        is_a = Space;
}
```

3. *Relations* which define a relation between a concept and another concept. In SPF-CL, relations are defined using a define Relation command. For example, the following statement defines a relation between doors and the Space classification.

```
define Relation {
        name = “related_space”;
        domain = Door;
        range = Space;
}
```

4. *Instances* which represent entities that exist physically or conceptually in the domain being modeled. In SPF-CL, concepts are similar to sets and instances are similar to elements of a set. For example, the following statements defines the instance "kitchenDoor" which is an element of the set of doors.

```
define Instance {
        name = "kitchenDoor";
        instance_of = "Door";
        related_space = "Kitchen";
}
```

As an example, the simplified version of the Clause-47.B from IMHZCode Figure 2.2 can be defined as shown in the next example.

```
define Limit {
        name = "IMHZCode#47#B#2;
        text = "Oda ve mutfak kapıları kasa dahil (0.90) metreden az olamaz."
        text_as_modeled = "Clear width of room and kitchen doors shall be at
        least 0.90 meters.";
        definition_type = Minimum;
        domain = instance roomDoor | kitchenDoor
        for_relation = relation width;
        range = 0.90 [meter]
}
```

The first field (name) in this definition is the section in which this clause is defined. In this case, this is Section 47.B.2 from IMHZCode. The second field (text) is the original text of the clause as defined in IMHZCode. The third field (text_as_modeled) represents an English explanation of this SML statement. The fourth field (definition_type) states that this provision defines a minimum limit. The fifth field (domain) states that this clause applies to only room and kitchen doors. The sixth field (for _relation) states that this provision limits the width of doors. The last field (range) states that the limit is 0.90 meters.

Fenves et al. (1995) claimed that by associating contexts with provisions, the need for large numbers of highly specialized subclasses from which to hang provisions is eliminated. However, while this model benefits from advantages of the combined approaches, it still inherits shortcomings from the object-oriented approach such as the problem of maintainability.

2.2.6. Semantic Models

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 important because building codes change continuously. To overcome the deficiencies of hard-coded representation approaches, recently, attention has been directed towards the study of semantic modeling approach, which is a relatively new method for knowledge representation.

The SMARTcodes project (AEC3, 2012) is a semantic approach which proposes to mark-up building codes in such a way that rules are dynamically generated in a computable format. SMARTcodes project provides a protocol and a software program (SMARTcodes Builder) for creating smart versions (tagged representations) of actual building code texts that reflects building codes with schema and tags used for automated compliance checking applications. 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. Figure 2.14 shows a block diagram of an illustrative system for generating SMARTcodes (Conover, 2009).

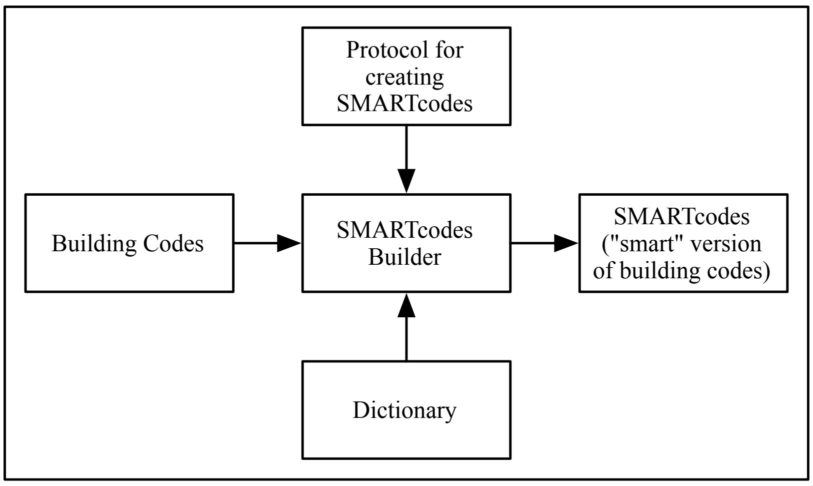

Protocol for creating SMARTcodes
Building Codes
SMARTcodes Builder
SMARTcodes ("smart" version of building codes)
Dictionary

Figure 2.14. SMARTcodes system architecture
(Source: Conover, 2009)

The system includes a SMARTcodes Builder, which is used along with a protocol to create tagged representations (smart versions) of the building codes. Building codes rely on dictionaries which are used by the SMARTcodes Builder pursuant to the protocol to facilitate creating the tags corresponding to a schema for SMARTcodes. The dictionaries may include term definitions, object model descriptions, data types, permissible units and operators. The dictionaries may further include schema information that correlates the SMARTcodes tags within the schema with the elements, units, operators and other information. Building codes must be converted into XML files before they are used in the SMARTcodes Builder. The SMARTcodes Builder receives building codes as XML and following the defined protocol, someone familiar with the building code creates SMARTcodes. A diagram of the protocol defined for creating SMARTcodes using the SMARTcodes Builder is given in Figure 2.15 (Conover, 2009).

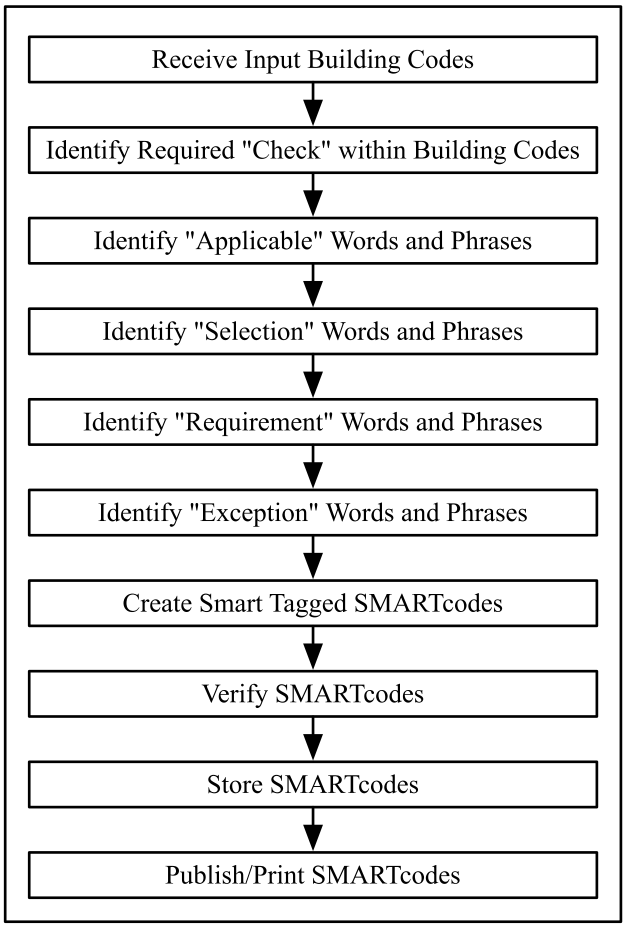
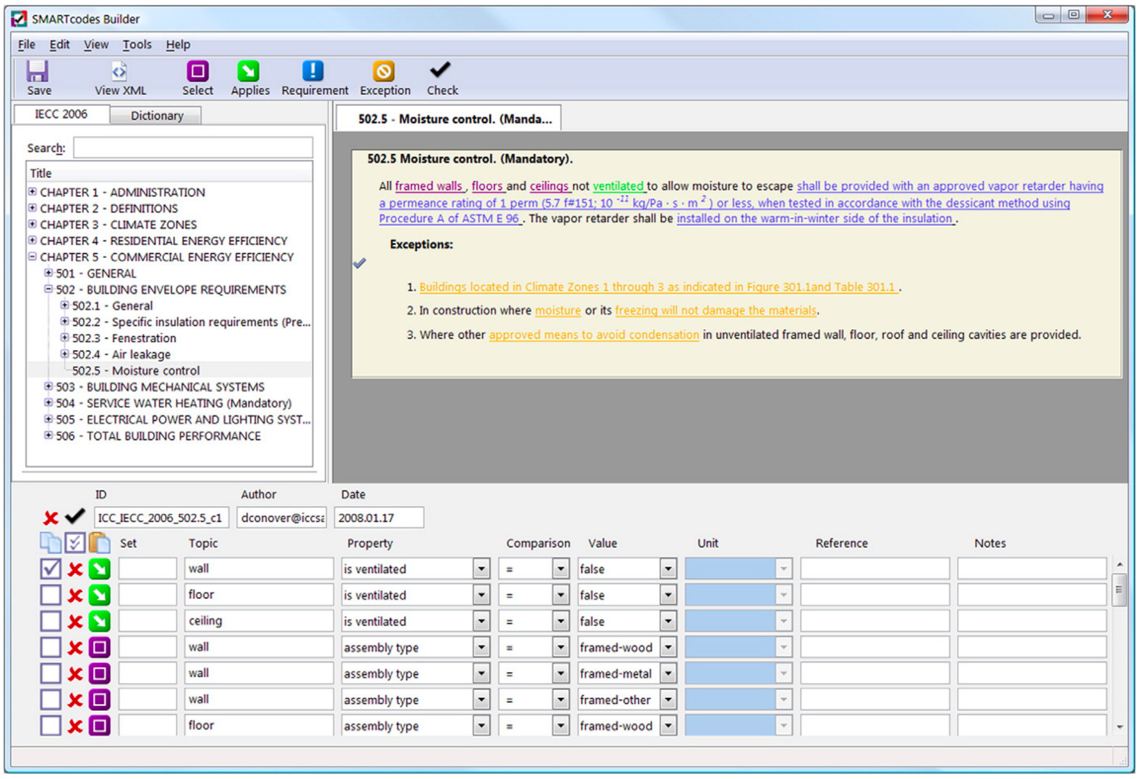

Receive Input Building Codes
Identify Required "Check" within Building Codes
Identify "Applicable" Words and Phrases
Identify "Selection" Words and Phrases
Identify "Requirement" Words and Phrases
Identify "Exception" Words and Phrases
Create Smart Tagged SMARTcodes
Verify SMARTcodes
Store SMARTcodes
Publish/Print SMARTcodes

Figure 2.15. Steps of the protocol for SMARTcodes
(Source: Conover, 2009)

Referring to Figure 2.15, in the first step, the user inputs the XML of the building code into the SMARTcodes Builder. In the second step, the user as guided by the protocol identifies the provisions in a specific building code section that gives rise to required checks using the SMARTcodes Builder interface. These checks correspond to a building code section or subsection that contains a specific requirement or related groups of requirements that must be applied to the building information model to determine whether the building design complies with the requirements. In the third step, the *applicability* words or phrases within the building code text associated with a check are identified and tagged. The applicability words or phrases are terms that each define a characteristic of the elements to which the check applies. In fourth and fifth steps, the *selection* and the *requirement* words or phrases within a check are respectively identified and tagged. In sixth step, the SMARTcodes Builder is used to identify any *exceptions* to the check, which are conditions under which the check is not applicable to the building model. In the seventh step, the tagged text is converted to xml according to the dictionary and schema. In this regard, the checks and identified applicability, selection, requirement, and exception atoms of the building code are coded according to the schema. In eighth step, the syntax within the SMARTcodes are checked to ensure functionality. In ninth step, the SMARTcodes are stored in a database. In the last step, the building codes represented by the SMARTcodes may be printed and published.

The SMARTcodes approach aims to enable non-programmers to define computable rules using simple tools. It is based on the RASE (Requirement, Applicability, Selection, Exception) methodology (Nisbet et al., 2009). 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. Each of these four constructs has attributes such as a property, a comparator and a target value with a unit. Code authors are able to markup these indicators that appear in the actual text of the code using SMARTcodes Builder software which creates the XML formatted version of the code. An example of RASE paradigm usage on a clause on moisture control from ICC IECC 2006 502.5 is shown as seen in the SMARTcodes Builder software in Figure 2.16 and in a short form in Figure 2.19. The moisture control rule *applies* to “framed” building elements. The rule does not cover all framed building elements. It is only for *selected* ones (“walls, floors, and ceilings") *except* "not ventilated" ones. The rule also has other exception conditions where this rule will not apply.


SMARTcodes Builder
File Edit View Tools Help
Save
View XML
Select
Applies
Requirement
Exception
Check
IECC 2006
Dictionary
Search:
Title
CHAPTER 1 - ADMINISTRATION
CHAPTER 2 - DEFINITIONS
CHAPTER 3 - CLIMATE ZONES
CHAPTER 4 - RESIDENTIAL ENERGY EFFICIENCY
CHAPTER 5 - COMMERCIAL ENERGY EFFICIENCY
501 - GENERAL
502 - BUILDING ENVELOPE REQUIREMENTS
502.1 - General
502.2 - Specific insulation requirements (Pre...
502.3 - Fenestration
502.4 - Air leakage
502.5 - Moisture control
503 - BUILDING MECHANICAL SYSTEMS
504 - SERVICE WATER HEATING (Mandatory)
505 - ELECTRICAL POWER AND LIGHTING SYST...
506 - TOTAL BUILDING PERFORMANCE
502.5 - Moisture control. (Manda...

502.5 Moisture control. (Mandatory).

All framed walls , floors and ceilings not ventilated to allow moisture to escape shall be provided with an approved vapor retarder having a permeance rating of 1 perm (5.7 f#151; 10 $^{-11}$ kg/Pa · s · m ^{2}) or less, when tested in accordance with the dessicant method using Procedure A of ASTM E 96 . The vapor retarder shall be installed on the warm-in-winter side of the insulation .

Exceptions:

1. Buildings located in Climate Zones 1 through 3 as indicated in Figure 301.1and Table 301.1 .
2. In construction where moisture or its freezing will not damage the materials.
3. Where other approved means to avoid condensation in unventilated framed wall, floor, roof and ceiling cavities are provided.

ID	Author	Date
ICC_IECC_2006_502.5_c1	dconover@iccsa	2008.01.17

	Set	Topic	Property	Comparison	Value	Unit	Reference	Notes
✓		wall	is ventilated	=	false			
		floor	is ventilated	=	false			
		ceiling	is ventilated	=	false			
		wall	assembly type	=	framed-wood			
		wall	assembly type	=	framed-metal			
		wall	assembly type	=	framed-other			
		floor	assembly type	=	framed-wood			

Figure 2.16. RASE constructs of a clause in the SMARTcodes Builder
(Source: AEC3, 2012)

ICC IECC 2006 502.5 Moisture control
All <*applicability*>framed</*applicability*> <*selection*>walls, floors</*selection*> and <*selection*>ceilings</*selection*> <*exception*>not ventilated</*exception*> to allow moisture to escape shall be provided with an <*requirement*>approved vapor retarder<*requirement*> having <*requirement*>a permeance rating of 1 perm</*requirement*> (5.7 × 10 –11 kg/Pa s m2) or less, when tested in accordance with the desiccant method using Procedure A of ASTM E 96. The vapor retarder shall be <*requirement*>installed on the warm-in-winter side</*requirement*> of the insulation. Exceptions: <*exception*>Buildings located in Climate Zones 1 through 3</*exception*>as indicated in Figure 301.1 and Table 301.1. In construction where <*exception*>moisture</*exception*>or its <*exception*>freezing</*exception*> will not damage the materials. Where other approved means to avoid <*exception*>condensation</*exception*>in unventilated framed wall, floor, roof and ceiling cavities.

Figure 2.17. RASE constructs of a clause in the short form

SMARTcodes project provided an authoring tool to manage the amendments of the building codes. SMARTcodes development stopped in 2010 but the underlying mark-up concept and RASE methodology used by SMARTcodes has been further developed by AEC3 Ltd. (Hjelseth, 2012).

2.2.7. Ontology-Based Models

Recently, the application of an ontology-based approach has been investigated as a possible computable framework for building code representation. Yurchyshyna et al. (2008) developed a formal ontology-based approach for the formalization and semantic organization of building codes. The research on formalization of building code texts is conducted through the following steps:

- Knowledge extraction from the texts of building codes into formal languages (e.g.XML, RDF) by analyzing the hierarchical structure of the documents and by adding new (meta)tags.
- Formalization of building codes by capitalizing the domain knowledge.
- Semantic mapping of building codes' content to existing knowledge bases, e.g. industry specific ontologies.
- Formalization of building codes in the context of the compliance checking problem.

In this work, in order to illustrate the feasibility of ontology-based modeling approach, a prototype is developed, which is called C3R (Conformity Checking in Construction with the help of Reasoning), that implements the algorithms of reasoning by expert rules according to organized conformity queries (Yurchyshyna & Zarli, 2009).

Ontology-based building code representation on the semantic web has also been explored by researchers (Pauwels et al., 2011). The research based on semantic web approach focuses on enhancing the IFC model by using description languages (e.g. SPARQL, Semantic Web RuleML (SWRL), Rule Interchange Format (RIF)) based on a logic theory (Pauwels et al., 2011). In their work, it is stated that IFC has several limitations when considering code compliance checking environments specifically such as limited expression range, difficulties in the partitioning of information and the possibility to describe the same information in numerous different ways. These limitations is mainly caused by the lack of a mathematically rigorous logic theory in the language deployed to specify the IFC schema. Pauwels stated that IFC's limitations can be overcome when deploying semantic web languages as an enhancement to IFC. By enhancing IFC onto a logical level, it could be possible to enable design and implementation of significantly improved code compliance checking systems.

There have been also some projects using semantic modeling approach and the application of industry specific taxonomies and ontologies in combination with Artificial Intelligence (AI) and Natural Language Processing (NLP) techniques to allow systems to interpret building code texts by automated or semi-automated data extraction (Cheng et al., 2009; Salama & El-Gohary, 2011; Zhang & El-Gohary, 2011). However, it would be quite challenging to come up with any automated method of information extraction because building code texts are written for human interpretation. Any building code representation should be in accordance with one set of the official interpretation provided by the authority.

2.3. Automated Compliance Checking

2.3.1. Introduction

Compliance checking is a process of building design project evaluation against all applicable rules of building code. It includes three sequential sub tasks:

- Gathering information about various aspects of the building design project.
- Comparing the design information with applicable rules of building codes.
- Documenting the results as evaluation reports.

Compliance checking is a complex, error prone and resource intensive process. Today, although every building project is modeled in a digital environment, these projects are checked manually for their compliance with building codes. Manual compliance checking may cause inconsistencies in approvals and results in delays for the overall construction process. Moreover, failure to correctly assess designs for compliance can have negative effects on building performance and allows errors that are expensive to fix. These potential problems are the main issues behind research on developing models for compliance checking process and developing automated compliance checking systems.

Automated compliance checking assesses a building design project on the basis of the configuration of building objects, their relations or attributes. Implementing a complete automated compliance checking system is a huge undertaking because these systems should have numerous complex functional capabilities. From the early efforts and the review of current work, Eastman et al. (2009) defined a necessary structure for implementing a functionally automated compliance checking system. Figure 2.18 illustrates the four classes of functionality an automated compliance checking system should support.

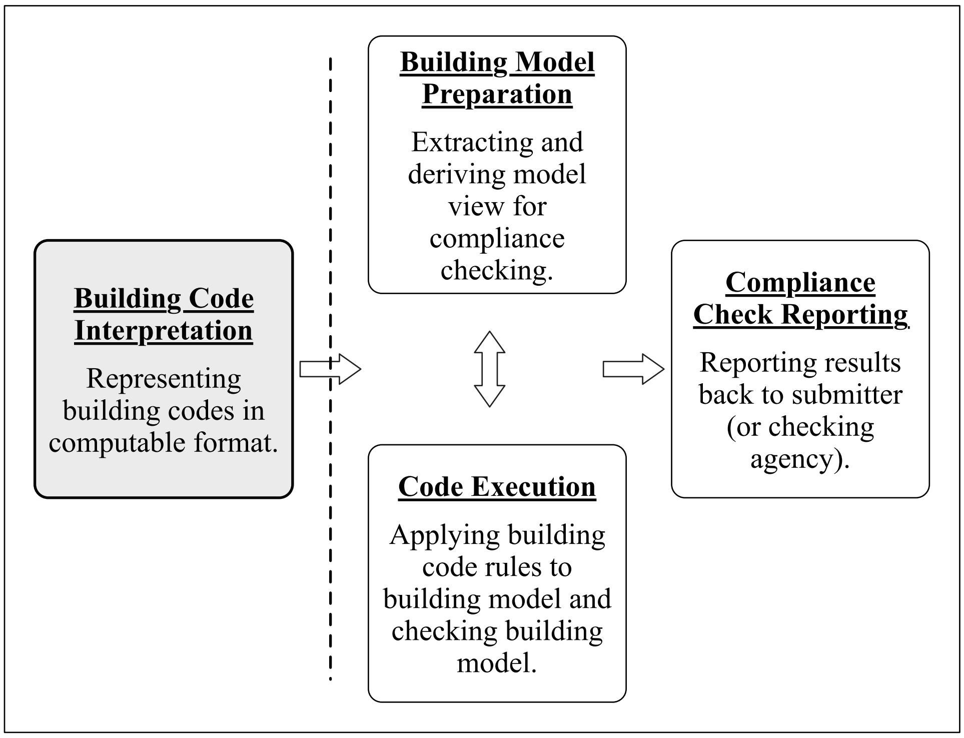

Building Code Interpretation
Representing building codes in computable format.
Building Model Preparation
Extracting and deriving model view for compliance checking.
Code Execution
Applying building code rules to building model and checking building model.
Compliance Check Reporting
Reporting results back to submitter (or checking agency).

Figure 2.18. The four classes of functionality a code compliance checking system

Automated compliance checking process is structured into four stages and it starts with building code interpretation. Building codes are written documents in human language formats which are only read and applied by people. For an automated compliance checking implementation, first, building code documents need to be represented in computable format. The second stage in this process is building model preparation. Building models are prepared by architects and designers but entering all required data for compliance checking cannot be expected from them. The preferred solution is to automatically extract related data and derive a model view to be checked. Initiatives such as those coordinated by buildingSMART (2008b) focus on this task. In the third stage, building code rules are applied to the derived building model view to check the level of compliance. Before rule execution, syntactic checking of the building model is required to determine if the building model carries all the data needed for checking. The last stage in compliance checking compiles the results and reports back to applicants and checking agencies.

After the brief explanation about compliance checking systems, major compliance checking projects is explained in the following section.

2.3.2. Automated Compliance Checking Systems

Research on the development of automated compliance checking systems for building models began nearly three decades ago (Garrett & Fenves, 1987), but efforts to produce working compliance checking systems came later. In this section, four major implementation efforts on automated compliance checking systems are examined. These efforts are CORENET, DesignCheck, SMARTcodes, and the effort by the General Services Administration in USA. Each of the four systems is summarized in Table 2.1.

Table 2.1. Overview of automated compliance checking systems

Development Agency	**Project**	**Year**	**Domain**	**Building Code Model**	**BIM Standard**	**Checking Platform**
Singapore	CORENET	1995	Zoning Accessibility Fire safety Environmental health Public housing Vehicle parking	Object based logic model	IFC	FORNAX EDM
Australia, CRC for CI	DesignCheck	2006	Accessibility	Object-oriented model	IFC	EDM
USA, ICC	SMARTcodes MCS	2006	Energy conservation	Semantic model	IFC	SMC
USA, GSA	DAT	2008	Circulation Security	Parametric table	IFC	SMC

2.3.2.1. Construction and Real Estate NETwork (CORENET)

CORENET is considered as a milestone effort. It is the first automated code compliance checking system for the AEC industry that has actually been used in practice. It was developed by the Singapore Building and Construction Authority in 1995 to automate the process through which building permits are acquired. CORENET is made up of 3 modules: e-Submission, e-PlanCheck, and e-Info. CORENET e-Submission module facilitates the accepting of building projects in digital format for automated checking and allows the tracking of the building permit acquisition process. CORENET e-PlanCheck is the main module that carries out the task of checking the building model against the building code. CORENET e-Info acts as a central repository and supplies various regulatory authorities involved in the process with reference materials in digital formats that can be regularly updated. Initially, the system was designed to work with 2D drawings, but since 1998 it relies on project data suplied by BIM systems in IFC format (Liebich et al., 2002). CORENET is able to check building projects for compliance with rules related to zoning, accessibility, fire safety, environmental health, public housing, and vehicle parking.

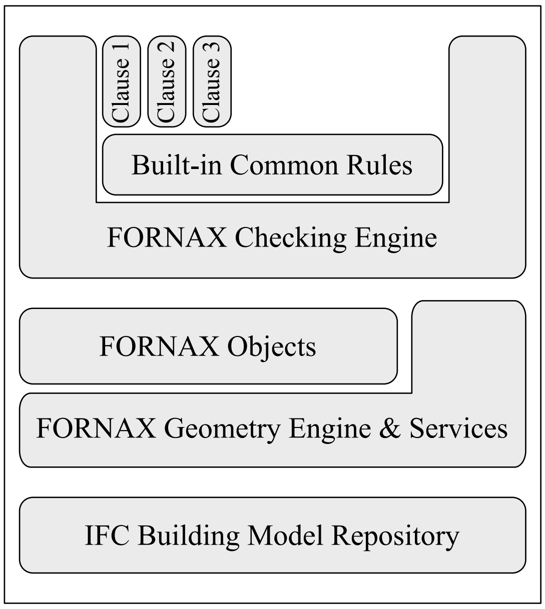


Figure 2.19. System architecture of CORENET project
(Source: novaCITYNETS, 2000)

In the CORENET project, the rules are embedded into the system. They are hard-coded by programmers. The system is built on top of the FORNAX platform that was developed specifically for this project (see section 2.3.3.1) FORNAX objects extend the IFC schema to add functionality for code checking. FORNAX objects extract the data required for code-checking from the building project that is in IFC format and the FORNAX checking engine applies the rules to these FORNAX objects and reports the results.

2.3.2.2. DesignCheck System

The DesignCheck project was initiated in 2006 by The Cooperative Research Centre for Construction Innovation based in Australia. The project was carried out in two stages. In the first stage, existing software platforms were evaluated in order to determine the technologies most suited to processing Australian standards. Two important software platforms, Solibri Model Checker (SMC) and Express Data Manager (EDM) were compared through tests that involved modeling actual code (initial code was Australian Standard AS1428.1 "Design for access and mobility"). The results of the testing stage indicated that EDM was comparatively more flexible in editing and writing of new rules. In the second stage, the DesignCheck system was developed (Ding et al., 2006).

The DesignCheck system is an automated code compliance checking system that enables compliance assessment against building codes. The system holds design information extended for code checking and encodes domain knowledge embedded in building codes. It defines an internal model based on IFCs for modeling the extended design information. This internal model allows for the definition of comprehensive building design information that corresponds to definitions that exist in building codes. In this system, the building code is interpreted using an object-based representation and then encoded into the EDM rule bases. Code-compliance checking (validation) of the internal model is carried out by EDM functions utilizing rule bases.

The architecture of the DesignCheck system is illustrated in Figure 2.20. It consists of three components: main user interface, EDM database and the report system. The main user interface enables monitoring the checking of designs, viewing check results. The EDM database contains building models, rules bases and the check results.

The building model is imported to the EDM database in IFC2x2 format and then mapped onto the DesignCheck internal model. The DesignCheck internal model is validated against rules in the rule bases. The results model stores the check results. The report system reads the check results from, and writes the comments to the results model in the EDM database. The report system provides both an interactive report page and a print friendly report page.

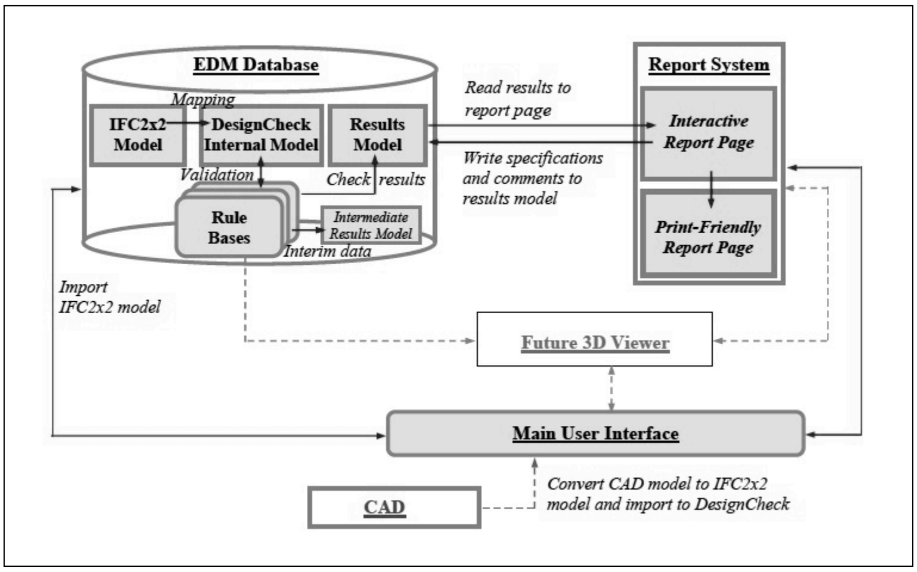

EDM Database
Mapping
IFC2x2 Model
DesignCheck Internal Model
Results Model
Validation
Check results
Rule Bases
Intermediate Results Model
Interim data
Read results to report page
Write specifications and comments to results model
Report System
Interactive Report Page
Print-Friendly Report Page
Import IFC2x2 model
Future 3D Viewer
Main User Interface
Convert CAD model to IFC2x2 model and import to DesignCheck
CAD

Figure 2.20. Architecture of the DesignCheck system
(Source: Ding et al., 2006)

2.3.2.3. SMARTcodes Model Checking System (MCS)

The SMARTcodes project was initiated in 2006 by AEC3 and Digital Alchemy with support from ICC. This U.S.A. based effort's main goal was to simplify the conversion of text based building codes into computable rule sets. SMARTcodes focused on automating and simplifying code-compliance checking of building projects (Conover, 2007). The system has initially implemented the International Energy Conservation Code (IECC). It utilizes the IECC dictionary for defining objects and properties for code checking. The computable rules are built using the *SMARTcodes builder* interface that is a web based application designed to reduce errors in interpreting the original code document. The IECC dictionary is used while building the rules and also facilitates the mapping between the system and the IFC model. Figure 2.21 illustrates the framework of the model checking system based on SMARTcodes.

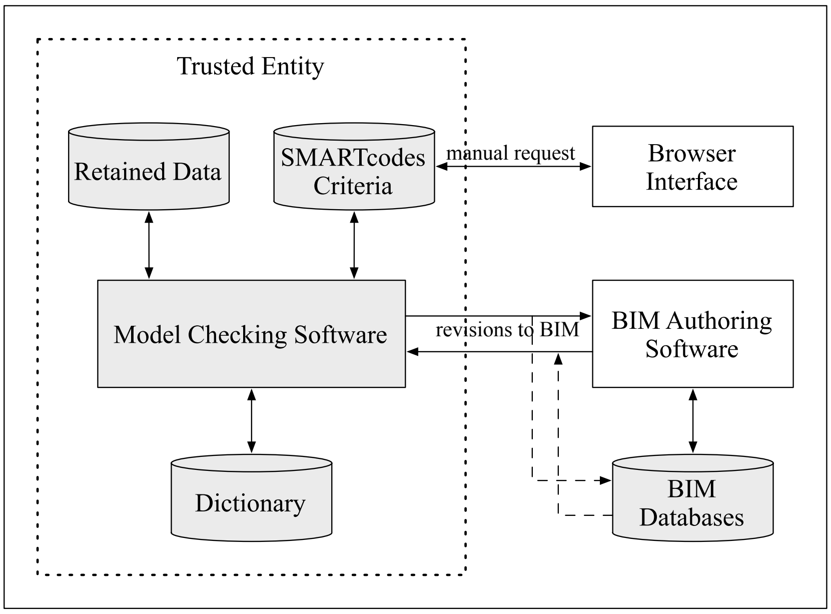


Figure 2.21. Framework of SMARTcodes model checking system
(Source: Conover, 2009)

2.3.2.4. Design Assessment Tool (DAT)

Design Assessment Tool (DAT) has been funded by the General Service Administration (GSA) and developed at the Georgia Institute of Technology in 2008 (Eastman et al., 2008). DAT is a rule checking system for circulation and security validation of U.S. courthouses. In this system circulation and security rules of the USA Courts Design Guide (CDG) have been modeled. The Solibri Model Checker platform is used. The rule statements were grouped by conditions and modeled as a set of parametric rules in SMC making use of a plug-in developed for this purpose. The building project in IFC format is designed using BIM tools, following the GSA Series Six BIM Guide for Circulation and Security Validation. The checking module utilizes a topological graph for modeling the relationship between spaces and a metric graph for representing movement paths and distances within spaces (Lee et al., 2010). Based on these two graphs, the checking module is able to assess if circulation paths between various spaces meet code requirements.

2.3.3. Technologies

Automated compliance checking systems are highly complex applications and they require significant software libraries to provide the needed functionality. Although efforts on developing automated compliance checking systems has been ongoing for nearly 20 years, existing platforms that support these systems is limited. FORNAX, EXPRESS Data Manager (EDM) and Solibri Model Checker (SMC) are three major platforms currently available that have been developed to support implementation aspects of compliance checking systems. These platforms provide object-based rule engines applying building code model rules to building information model (BIM) data. All of these platforms utilize the Industry Foundation Classes (IFC) as the neutral building information model file format. In the next sections, these platforms for compliance checking systems will be explained.

2.3.3.1. FORNAX

FORNAX is the first large effort in code compliance checking. It was developed by novaCITYNETS Pte. Ltd, an e-Government solution provider in Singapore, on top of EDM Model Checker (novaCITYNETS, 2000). FORNAX uses the basic building model information, exchanged in IFC format, and adds to it the required missing information associated with the building code compliance checking procedures. FORNAX platform has a C++ object library that takes necessary BIM data from the IFC file, derives the needed data for code compliance checking, and generates extended views of IFC data. FORNAX objects carry rules for assessing themselves. FORNAX objects are extendable to accommodate attributes and methods required to check the building project for code compliance. FORNAX is essentially developed as a development and deployment platform for the CORENET project. In this project, building codes are modeled on top of the FORNAX platform. FORNAX has since been considered by a number of other building code compliance checking efforts as a possible platform (Khemlani, 2005).

2.3.3.2. EXPRESS Data Manager (EDM)

EXPRESS Data Manager (EDM) is a comprehensive suite of model-driven database systems developed by Jotne EPM Technology, a Norwegian IT company (Jotne, 1994). EDM has been built specifically to manage the extremely complex data structures found in industrial applications that deal with what is generally known as product data models. EDM implements the methodology of the open international standard IS0 10303, commonly referred to as STEP (the STandard for the Exchange of Product Model data). EDM started out as a collaboration tool supporting work processes for data exchange, data sharing, data integration, and data archiving, but it has continued as a comprehensive open development environment with functional software tools using the EXPRESS language. The EDM package provides several additional modules including EDMmodelServer and EDMmodelChecker. EDMmodelServer module is an object-based database server providing textual reporting and server services. It consists of an object-based database, and a standardized set of tools to interact with the data (STEP or IFC) in the database. EDMmodelChecker is a model checking tool supporting the open development of EXRESS-based rule checking. This EDM module is open to user extensions. It allows users to build new computable rules using EXPRESS-X and provides an environment to execute them. These EDM modules are complex and require a high level of expertise in EXPRESS language to make use of them (Eastman et al., 2009). Several code compliance checking efforts used the EDM as part of their implementation, including the CORENET and DesignCheck projects.

2.3.3.3. Solibri Model Checker (SMC)

Solibri Model Checker (SMC) is an object-based stand-alone platform application developed by Solibri, a Finnish software company (Solibri, 1999). SMC provides a model checking tool that reads a BIM model, in IFC format, and maps it to an internal structure facilitating access and processing. It is capable of directly interfacing with BIM systems, visualizing building models, clash checking and code compliance checking against rule sets. The checking is carried out using parametric rules. Users can change the parameters of certain rules according to the building code.

SMC offers a set of built-in rules, and a rule configuration based on parameters. It also has a built-in tool for developing of rule sets, called Constraint Set Manager (CSM). New rules are added in java using the CSM. However, CSM is not publicly available, restricting the rules to be checked to those supplied by Solibri. This means that users are not capable of editing or modifying built-in rules. Nor are they able to add new rules without editing the original code.

Industry Foundation Classes (IFC):

Industry Foundation Classes (IFC) is a standard building data model specification developed in 1994 by the International Alliance for Interoperability (IAI) that was later renamed as buildingSMART in 2005. IFC was developed specifically to meet the needs of the AEC industry (buildingSMART, 2008a). The IFC specification is a neutral data format for the representation of building information to enable interoperability between different software systems within the AEC industry.

The IFC specification (ISO 16739) is derived from the STEP standard (ISO 10303) which is an open international standard for the representation and exchange of product data in all industries. The IFC specification is written using the EXPRESS language that is a data definition language specified in part 11 of the STEP standard (ISO10303-11). The IFC exchange file format (.ifc) uses the "STEP physical file" format defined as ISO10303-21. In addition to the IFC_EXRESS specification there is an ifcXML specification to represent IFC building data model using XML schema specifications. The ifcXML exchange file format (.ifcXML) is the XML document structure. IFC also specifies techniques for extending the schema such as the IFC Property Set (PSet) that is a mechanism for adding custom properties to the standard IFC schema. Substantial efforts have been made to continuously develop the IFC specification. The current version of IFC was released in 2013 as IFC4. IFC is widely used and there are many reference sources on the IFC (ISO, 2002; Liebich, 2002; ISO, 2004; Khemlani, 2004; Eastman, 2006; buildingSMART, 2008a; ISO, 2013).

IFC is regarded as the dominant option for building information modeling in the AEC industry. Major BIM authoring software tools such as Autodesk Revit, Graphisoft Archicad, and Bentley MicroStation support IFC and have the capability of exporting their own building model data in IFC format. Several automated code compliance checking efforts utilized IFC as a neutral file format (Han et al., 1998; Sing & Zhong, 2001; Yang & Li, 2001; Ding et al., 2006; Wix & Conover, 2007; Eastman et al., 2008).

CHAPTER 3

BUILDING CODE REPRESENTATION AND MODELING METHODOLOGY

3.1. Introduction

This thesis presents a new representation model and modeling methodology for building codes to support the development of compliance checking systems. The new representation model adopts the four level representation paradigm as a theoretical base and uses the semantic modeling approach for developing the building code representation. The new model organizes the representation in 4 levels. The tasks at each of the levels are (bottom-up): 1) modeling domain concepts by defining *domain objects* to represent concepts that exist in the building code text, 2) representing individual rule statements by employing formal computable semantic *rule objects*, 3) structuring dependency relations between the rule objects and thus defining *rule-set objects*, and 4) representing the organization of building codes via the creation of *management objects* that categorizes closely related rule-sets.

3.2. Four Level Representation

The new model is based on the four level representation paradigm (Fenves & Wright, 1977) which is derived on an abstract model of the logical structure of building codes identified by Nyman and Fenves (1975). According to this general structure, the content of the building code is described in 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 models. Nyman's research provides a solid foundation for modeling building codes and defining the process of modeling building codes. This general organizational structure was initially used for the SASE model as a representation framework by Fenves et al. (1987) Subsequent studies on building code modeling are also based on this approach (Jain et al., 1989; Rasdorf & Lakmazaheri, 1990). Although this approach was based on a solid theoretical foundation, its application in the field was not practical and models based on this theory were not widely adopted in AEC industry. Literature review reveals two important reasons for this failure. First reason is the lack of domain models that identify domain specific objects with their attributes and relationships. The lack of an industry standard in building modeling led to a high number of idiosyncratic data item definitions and increased the complexity of the building code models (Hakim & Garrett, 1992). Second reason is related to the representation methods used in the modeling of building code information. Decision tables and programming languages used for representation quickly became hard to maintain and build. The complexities involved in actual building codes proved too difficult for these methods of representation (Fenves et al., 1995). In summary, the four level paradigm was not adopted in practice, mainly due to the fact that information technologies for knowledge representation were not mature enough at the time.

The new model developed for this dissertation adopts the four level paradigm as a theoretical base for representing building codes but addresses the above issues on knowledge representation methods by utilizing the relatively recent semantic modeling method that will be discussed in the next section.

3.3. Semantic Representation – RASE Model

Building codes have been represented in a number of ways, e.g. as decision tables, hard-coded rules, programming logic, domain-specific rule language etc. Ideally the representation should be independent of compliance checking systems. It should also be adaptable to continuing building code amendments. The key is to make it possible for domain experts, who generally do not have any programming knowledge to manage the representations themselves. Semantic modeling approach, which is a relatively new method for knowledge representation, aims to meet the above requirements. One recent project, SMARTcodes, can be referenced as a good example of the use of this approach (Conover, 2009). SMARTcodes project aims to define computable rules using simple tools that enable non-programmers to create representations by tagging actual building code texts. It is based on the RASE (Requirement, Applicability, Selection, Exception) methodology. RASE defines four common constructs that make up a rule. These constructs are used to identify the building code essence from the actual text of the code (Nisbet et al., 2009).

RASE paradigm 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. 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 the building elements. RASE paradigm utilizes these four types of indicators as a basis of the common constructs of checks. Each of these four constructs has the following attributes: Topic, property, comparator, value and unit. Building code authors are able to markup these indicators that appear in the actual text of the code using SMARTcodes Builder software which creates an XML formatted version of the code (Conover, 2009).

RASE paradigm is a good starting point for modeling building codes. It provides an easy to understand simple method for deconstructing rule sentences. It also accommodates a scheme where code authors are able to build and maintain building code representations. In order to evaluate the capability of the RASE method to model real building codes, a pilot study was conducted for developing a representation for the Izmir Municipality Housing and Zoning Code (IMHZCode) using RASE constructs (Macit et al., 2013). IMHZCode rules have been modeled as requirement, applicability, selection, and exception objects. Some examples of how rules are modeled are shown in

Table 3.1. The experiences gained in this study showed that RASE methodology offers an ease of use for non-programmers and could be adapted to represent IMHZCode, however, some modifications would be beneficial to overcome three shortcomings that were identified.

Table 3.1. Examples of rules from IMHZcode modeled according to the RASE

Id	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= duplexHouses	pitch<=33%

First shortcoming is the unnecessary repetitions that occur due to the independent modeling of individual rule statements. Representations of the same concepts, referenced by multiple rules, are repeated many times for each *applicability* or *selection* construct they are a part of. While this approach allows rules to be independently modeled, it is prone to inconsistencies and creates redundancies. Inconsistencies may develop when the same concept is modeled differently in different contexts (rules). Building codes, in general, have a number of rule statements that indicate different *requirements* about the same concept. This is true especially for IMHZCode. Hence, the pilot study results unveiled a high number of redundant definitions especially for applicability constructs. The Domain Level that will be discussed in the next section is introduced to address this issue by creating a lower level library that can be used to define these repeating concepts once.

Second shortcoming is the lack of explicit relationships between individual rule statements. RASE methodology delegates that responsibility to processing at a higher level within automated compliance checking systems. In the SMARTcodes project, first the original code text is marked-up, then this marked-up text is structured into an XML representation, and in the last stage the XML file is used to create computable rules for the automated checking system. It is in this final stage that the relationships are represented in the form of an IFC constraint hierarchy. This totally independent handling of rule statements simplifies deconstructing rules for especially non-programmer code authors. However, with the relationship representation taking place separately in the automated checking system, it becomes impossible to ensure correctness and consistency for the overall code representation independent of automated checking systems. Moreover, this split organization has a negative impact on the maintainability of the representation as well. It is necessary to represent the relationships independent of automated checking systems. The *Management Level* and the *Rule-set Level* of the new model are introduced to address this shortcoming and they will be discussed in the following section.

The third shortcoming is about the *exception* construct of RASE. The pilot study revealed that it is unnecessary to represent separate *selection* and *exception* information for individual rule statements. Exception information is the opposite of selection information and it can be represented within the *selection* construct by including a "excludes" comparator. In the new model, the exception constructs are eliminated.

In summary, while the RASE methodology was adopted, it was modified into a four level representation that improves it by eliminating redundancies and adding logical relationships. It is possible to ensure the conciseness and consistency of not just individual rule statements but the overall code model. The new model is completely independent of actual checking systems and thus should be easier to maintain. The new model is discussed in detail in the next section.

3.4. Building Code Representation

The new building code representation model developed as part of this thesis combines the semantic modeling method established by the SMARTcodes project (Wix, 2008) with the theoretical foundations set by Nyman and Fenves (1975), namely the four level methodology. This new hybrid approach aims to;

1. establish a building code representation *independent* of checking systems,
2. preserve the high level of *maintainability* in RASE,
3. minimize redundancies by introducing a hierarchical structure across four levels and improve on *conciseness*,
4. offer a level in which rule relationships are modeled and monitored so that *consistency* of a building code model can be ensured.

The new building code representation model is proposed to provide a systematic structure for representing building codes in computer implementable format. The proposed model consists of four levels:

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.

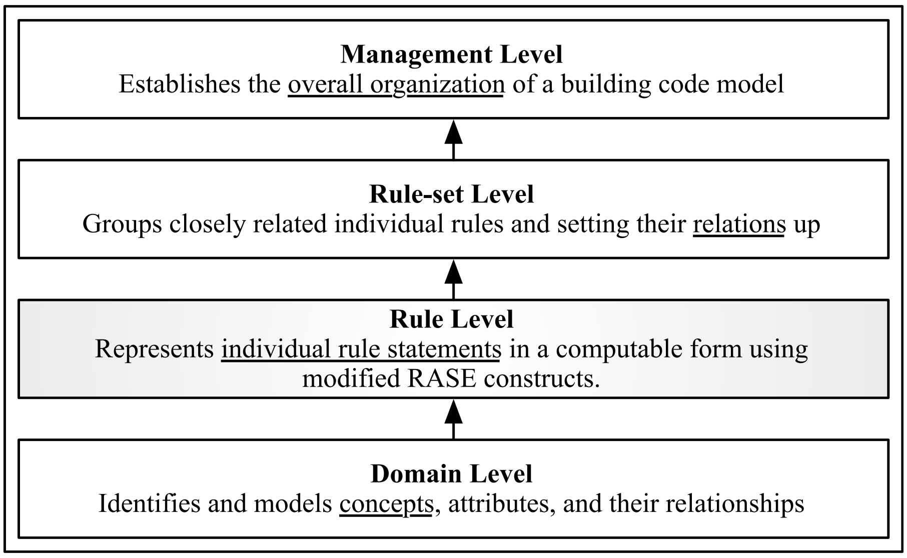

Management Level
Establishes the overall organization of a building code model
Rule-set Level
Groups closely related individual rules and setting their relations up
Rule Level
Represents individual rule statements in a computable form using modified RASE constructs.
Domain Level
Identifies and models concepts, attributes, and their relationships

Figure 3.1. Four level structure of the new model

3.4.1. Domain Level

Building codes state requirements to be met, but also describe objects subject to these constraints. Building codes should be modeled in a way that reflects this dual purpose. Modeling rules and constraints are about "how" an entity materializes while modeling entities is about "what" the entity is. The two naturally have differences in representation.

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.). Automated compliance checking systems assess building projects after mapping these concepts and entities in the build code representation to objects that constitute a building project. The mapping process will benefit from modeling of these domain specific concepts and entities independent of the rules with hierarchical relationships that are similar to the ones in the building information model.

RASE methodology integrates the code requirements and the domain specific concepts and entities in a single rule representation. While this simplifies conversion of building code texts into a computable building code model, it requires defining the same concepts and entities multiple times for every rule where they are referenced. This may lead to inconsistencies especially when the concept or entity definitions require updates.

By first defining the domain specific concepts and entities, independent of the rules, a domain model is created. The domain model acts as a library of objects that are utilized during the modeling of individual rule statements. The library objects can be used as building blocks during modeling and maintenance of the rules by code authors with no programming background.

This domain model also helps expose how domain specific information can map to building models in external systems. It is aimed at facilitating communication and interoperability among building information models, building code models, and automated compliance checking systems.

For the new model, creating domain objects that form the lowest level is naturally the initial step. In this step, concepts and entities referenced in the building code text are identified and modeled with their attributes and their relationships to each other. The output of this level is a domain model that is a library of all required concepts and entities with attributes and relationships. These building code domain objects are used when modeling rules. The applicability constructs in RASE methodology are filled by selecting from this library of domain objects.

3.4.2. Rule Level

Representing the individual rule statements in computable format is the second step in the new model. 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. These modeled rules are later used to check compliance of projects that are also in digital form.

In this *rule level*, individual rule statements are represented as rule objects in the form of structured data based on the modified RASE constructs. Rule statement semantics of the building code are captured in rule objects. Each rule object determines a single requirement for a specific attribute of objects that meet specific criteria. Every requirement is specified by a specific value and a method for comparison. All objects reside in the domain model.

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 object and may have zero or more selection objects. Basic structure of the rule object is given in Figure 3.2

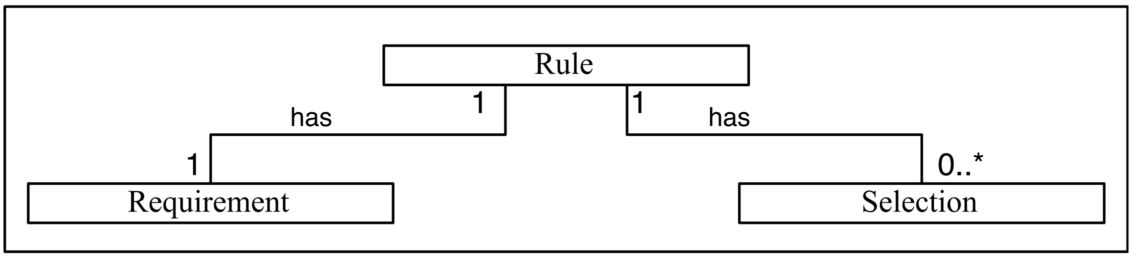


Figure 3.2. Basic structure of the rule object

Selections define the circumstances under which a rule is applicable. The modeling of selection criteria as part of a rule allows all rules to be applied during the automated checking process and eliminates the possibility of unfired rules in rule based systems. To check for conformance with a rule, its requirement is checked. If the selection under which a rule is applicable is true, the requirement is checked for conformance. If not, the rule is regarded as inapplicable and ignored.

Both requirement and selection objects have the general form: A "subject" and a "predicate" (Figure 3.3).

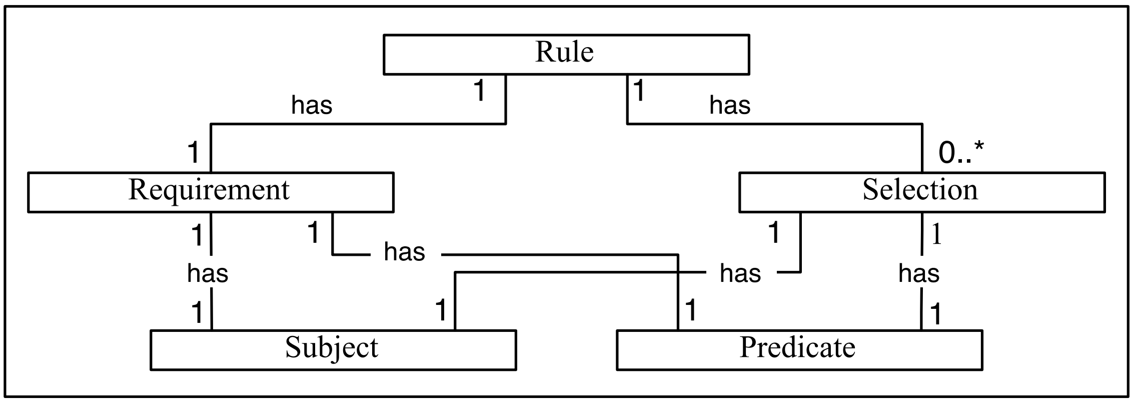


Figure 3.3. Structure of the rule object accommodating properties

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. (Figure 3.4)

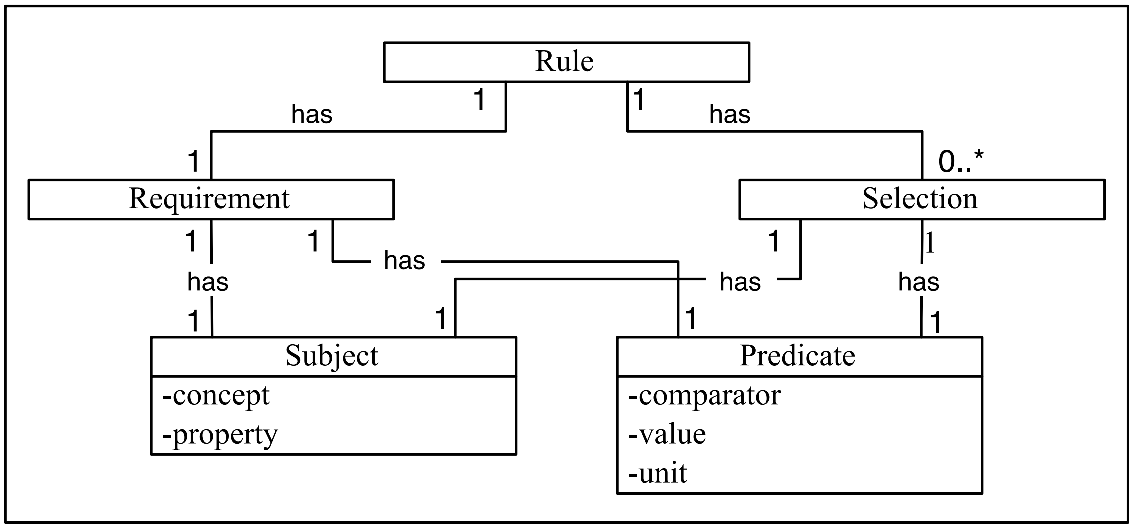


Figure 3.4. Detailed structure of the rule object

In the modified RASE, rules appear to be made up of only requirement and selection constructs. However, the applicability and exception information also exist. Applicability information is embedded as the subject part of the requirement objects and exceptions are handled within the selection objects.

When building a rule object, the concept and property will be drawn from the library of concepts modeled in the lower *domain level*. The concept list would be editable for adding new concepts when the building code is revised.

Building rule objects that form the *rule level* of the representation is the second step of the overall modeling process. At the end of this step, all rule statements in the code are modeled as separate objects that are ready to be linked at the higher level.

Table 3.2 and Figure 3.5 illustrate examples of the rule representation in both table and XML formats.

Table 3.2. An example of rule representation in table form

Requirement					**Selection**				
concept	**property**	**comparator**	**value**	**unit**	**concept**	**property**	**comparator**	**value**	**unit**
door	height	≥	210	cm	-	-	-	-	-
door	width	≥	130	cm	door	relatedSpace	equal	mainEntrance	-
door	width	≥	100	cm	door	relatedSpace	equal	entrance	-
door	width	≥	90	cm	door	relatedSpace	equal	room	-
door	width	≥	90	cm	door	relatedSpace	equal	kitchen	-
door	width	≥	80	cm	door	relatedSpace	equal	bathroom	-
door	width	≥	80	cm	door	relatedSpace	equal	cellar	-

```
<Rule 1>
        <Requirement>
                <Subject>
                        <concept> door </concept>
                        <property> height </property>
                </Subject>
                <Predicate>
                           <comparator> ≥ </comparator>
                           <value> 210 </value>
                           <unit> cm </unit>
                </Predicate>
        </Requirement>
</Rule 1>
<Rule 2>
        <Requirement>
                <Subject>
                        <concept> door </concept>
                        <property> width </property>
                </Subject>
                <Predicate>
                           <comparator> ≥ </comparator>
                           <value> 130 </value>
                           <unit> cm </unit>
                </Predicate>
        </Requirement>
        <Selection>
                <Subject>
                        <concept> door </concept>
                        <property> relatedSpace </property>
                </Subject>
                <Predicate>
                           <comparator> equal </comparator>
                           <value> mainEntrance </value>
                </Predicate>
        </Selection>
</Rule 2>
<Rule 3>
        <Requirement>
                <Subject>
                        <concept> door </concept>
                        <property> width </property>
                </Subject>
                <Predicate>
                           <comparator> ≥ </comparator>
                           <value> 100 </value>
                           <unit> cm </unit>
                </Predicate>
        </Requirement>
        <Selection>
                <Subject>
                        <concept> door </concept>
                        <property> relatedSpace </property>
                </Subject>
                <Predicate>
                           <comparator> equal </comparator>
                           <value> entrance </value>
                </Predicate>
        </Selection>
</Rule 3>
```

Figure 3.5. An example of rule representation in XML format

3.4.3. Rule-set Level

Defining the relationships between rule objects representing individual rule statements is the third step in the new model. In the *rule level* (second step) 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. Rules can be joined with an OR conjunction when modifying the requirements and an AND conjunction when adding new requirements.

It is important to identify how these rule objects combine. Rule objects are related to one another through the property addressed. Rule objects are typically cumulative. If there are several rule objects that indicate various requirements to be met by a particular property of the same concept, it is expected that all of these rules be satisfied. If rule objects make an exception or modification to the requirement of other rules, then these rules are alternatives. Only one of these rules will be applicable.

In the new model related rule objects 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. Figure 3.6 shows the structure of the rule-set object.

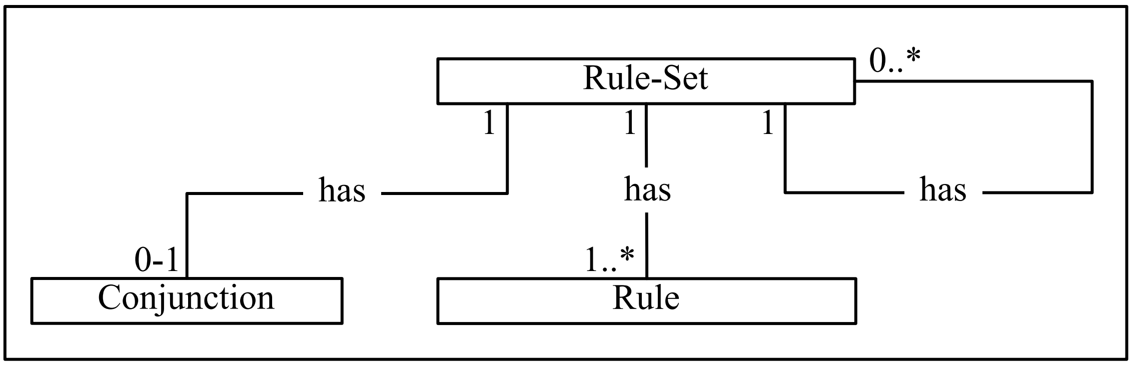


Figure 3.6. Structure of the rule-set object

3.4.4. Management Level

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. An organizational system can also be used to develop an outline to arrange the rules and to define the scope for the building code. In the new model, the fourth step addresses issues related to the organization of the rule-sets modeled in the *rule-set level* (third step).

One natural method of organization is the one that reflects the original building code text. Rule-sets can be grouped to reflect the clauses and clauses can be ordered under sections following the order in the actual code document. However, it is beneficial to allow for alternative organization schemes to exist simultaneously. One such alternative organization is to group rule-sets according to the concepts they impose requirements for. In the actual code document requirements on a single concept can span across multiple clauses making it difficult to recognize inconsistencies. To overcome this shortcoming a concept-based organization is preferable. The concept-based organization is also helpful for automated compliance checking algorithms in identifying all rules that need to be processed for a given concept. The *management level* is included to allow alternative networks of linked rule-sets to co-exist. Moreover, this top level of the new model allows various building codes (such as Fire Safety Code, High rise code) to be aggregated exposing possible conflicting provisions on concepts.

These organizing networks are modeled using rule-set group objects that form a tree. The root of the tree represents to overall code while the leaves are the rule-sets that are defined at the third level. Any number of intermediary nodes can be defined and they represent headings and sub-headings (sections and clauses in the actual document organization). Figure 3.7 shows the structure of the rule-set group object.

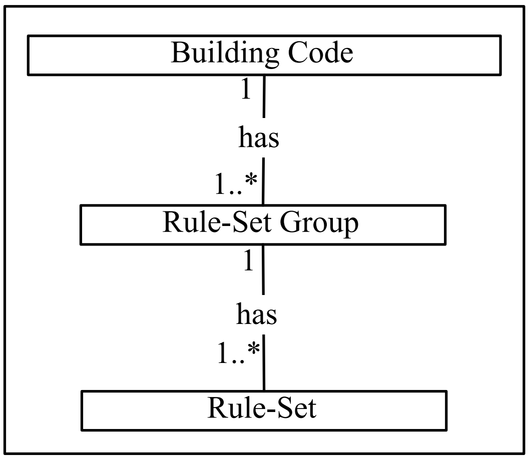
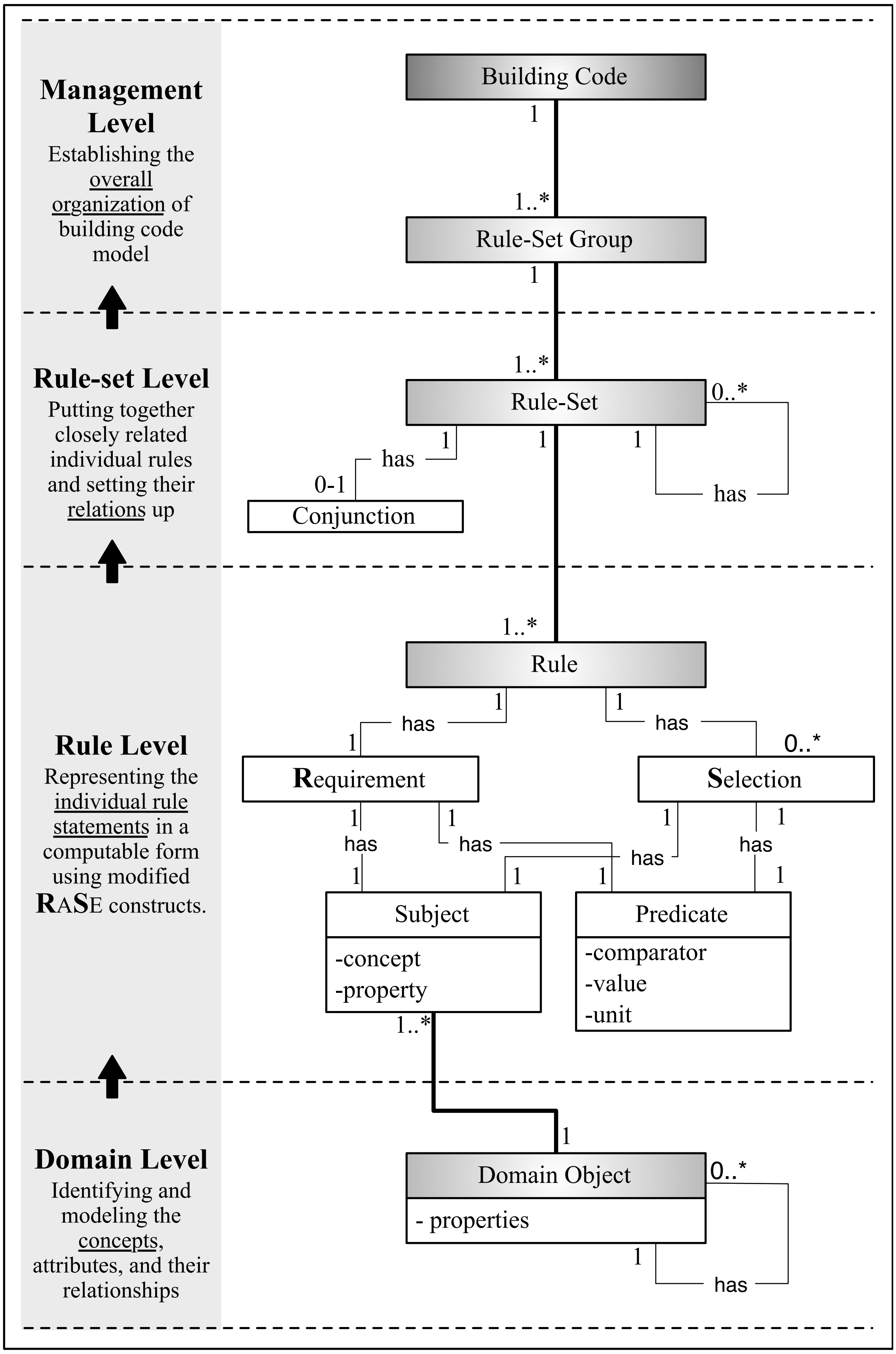


Figure 3.7. Structure of the rule-set group object



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

Figure 3.8. Overall structure of building code representation model

3.5. Building Code Modeling Methodology

The modeling process is crucial in order to develop building code representations. Representing building codes in computer implementable format is not only a technological issue, but also a process issue. Developing building code representations by utilizing the proposed model requires a clear, transparent, and well defined process.

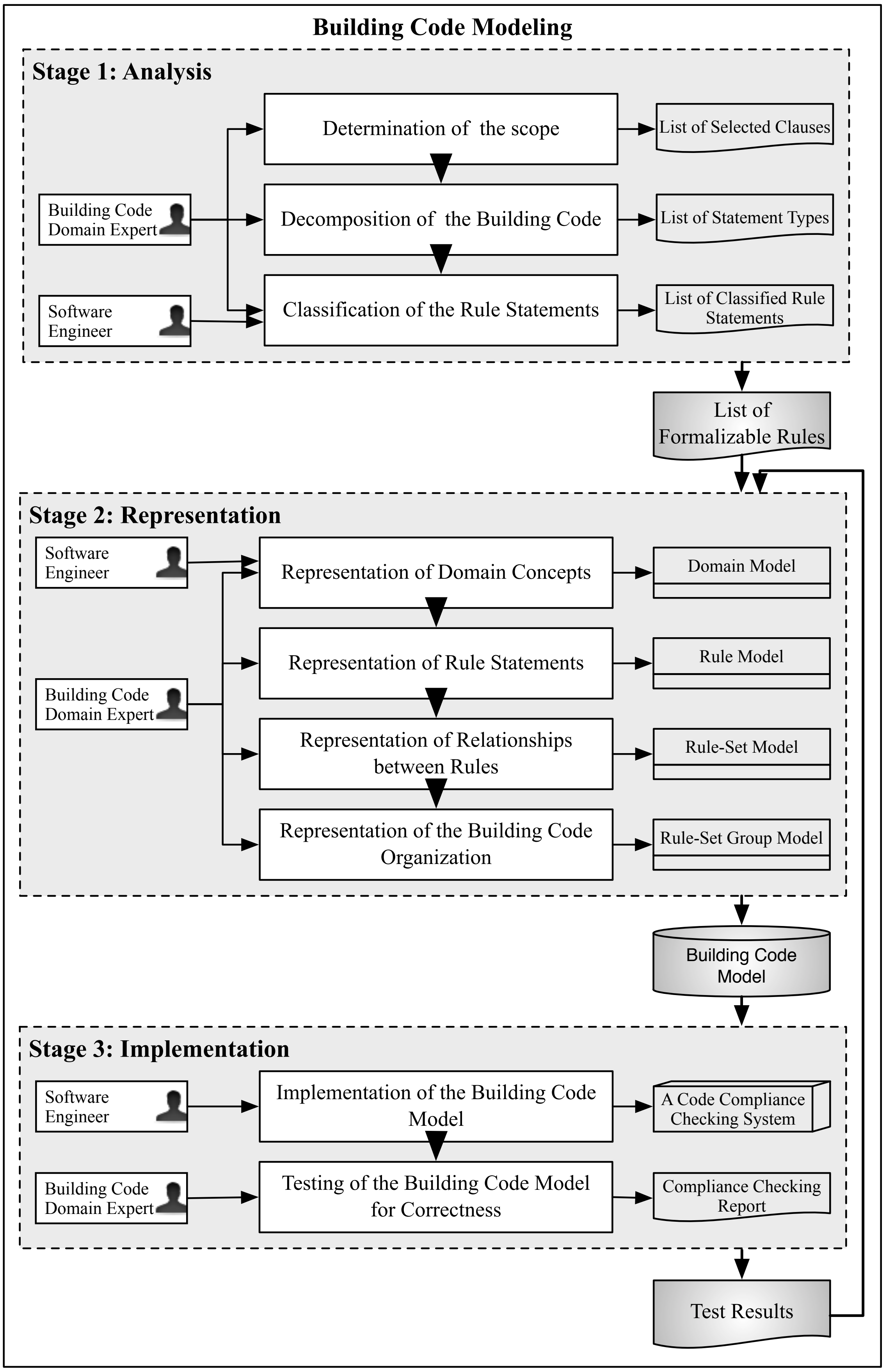
A straightforward methodology, which comprises three process stages for building code representations, is proposed in this thesis. Below are the recommended process stages to develop building code representations based on the proposed model. Each stage identifies what should be done, what should be delivered from this stage, and who the main actors in this step should be.

Stage 1: Analysis of the building code to define what should be represented explicitly for the purposes of automated compliance checking and to document how much of the building code can be modeled reliably.

Stage 2: Representation of the building code by utilizing the developed representation model.

Stage 3: Implementation of the building code model within a compliance checking application.

Next section explains these stages in detail. Figure 3.9 illustrates the stages of the building code representation methodology.



Building Code Modeling
Stage 1: Analysis
Building Code Domain Expert
Software Engineer
Determination of the scope
Decomposition of the Building Code
Classification of the Rule Statements
List of Selected Clauses
List of Statement Types
List of Classified Rule Statements
List of Formalizable Rules
Stage 2: Representation
Software Engineer
Building Code Domain Expert
Representation of Domain Concepts
Representation of Rule Statements
Representation of Relationships between Rules
Representation of the Building Code Organization
Domain Model
Rule Model
Rule-Set Model
Rule-Set Group Model
Building Code Model
Stage 3: Implementation
Software Engineer
Building Code Domain Expert
Implementation of the Building Code Model
Testing of the Building Code Model for Correctness
A Code Compliance Checking System
Compliance Checking Report
Test Results

Figure 3.9. Stages of the methodology for building code representation

3.5.1. Analysis Stage

The process of representing building codes in computer implementable format is not trivial due to the complex nature of building codes. It is essential to document the various types of information contained in building codes as well as the organization of the codes in order to develop a building code representation. Thus, analysis of the building code is the first stage in the proposed building code modeling methodology. This stage covers the following steps:

1. Determination of scope
2. Decomposition of the building code
3. Classification of the rule statements

3.5.1.1. Determination of the Scope

The first step in the analysis stage is to determine the scope. It should be clearly specified which building code, chapter, and clauses will be included in the representation. The assumptions and the scope of the building code representation as well as the goals for the work need to be defined leaving no ambiguities, gray areas or imprecise notions.

Building codes are complex written documents and they include various types of information. They are composed of different types of clauses. While some clauses define constraints about buildings, land readjustment, or construction issues others describe the general issues such as aim, scope and legal basis of the building code, or explain specific names used in the building code and give detailed definitions for various terms. Since the development of building code representations is aimed to be a base for automated code compliance checking, the building code clauses pertinent to buildings should be determined and documented. The output from this step will be a human readable document listing the parts of the code to be modeled, defining the scope of the representation. The actors of this step are the building code domain experts appointed by the authority.

3.5.1.2. Decomposition of the Building Code

The second step of the analysis stage is the decomposition of the building code. In this step, the clauses related to buildings are decomposed into a list of statements and all statement types that exist are determined. As explained in detail in section 2.1.2 building code clauses compose of different types of statements. While some of these statements are informative such as clarifications or applicability conditions, others relate to rules which all building projects must satisfy. Automated code compliance checking systems apply rules to a proposed project, therefore the statements defining rules are of interest. Decomposition of the building code should identify the various types of statements and extract the rule statements. The output of this step will be a human readable document listing statements with determined types. The actors of this step are the building code domain experts appointed by the authority.

3.5.1.3. Classification of the Rule Statements

In the last step of the analysis stage, rule statements are classified in order to document how much of the code as well as which types of rules can be modeled. Building codes may include rules that are open to interpretation, uncertain, sometimes even contradictory and impossible for modeling. It is needed to document how much of the code can benefit from automated code compliance checking. Classification of rules according to their formalizability will help to assess potential coverage of building code representations. In addition to the formalizability issue, building codes have a complicated structure. They contain closely related rules that are making exceptions, modifications, or clarifications to other rules as well as stand-alone rules that are unrelated to other rules. It is important to understand the relationship between rules in order to model them correctly. Classification of rules according to their self-containedness will help to figure out relationships between them. The output of this step is a list of rules that can be represented in computer implementable format. This step needs interdisciplinary knowledge in determining which concepts can be represented in computer implementable format. Cross-disciplinary collaboration should be provided and the building code domain experts should work with software engineers.

Analyzing the complex structure of building codes and determining different types of rules is the first stage in development of building code representations. During the analysis stage, the building code, the chapters of the code, and the clauses of the chapters to be represented are determined and all rules in selected clauses are classified. Next stage in the proposed building code modeling methodology is to represent the code by utilizing the developed representation model.

3.5.2. Representation Stage

Representation of the building code is the second stage in the proposed building code modeling methodology. In this stage that focuses on modeling, one important concern is about the structure of the building code representation. In the literature there are two approaches about how the structure of the building code representation should be. Han et al. (1998) suggests that the structure of the building code representation should be similar to the structure of the building information model. On the other hand, Nisbet et al. (2009) believes the structure of the building code representation should be similar to the structure of the building code. While Han's approach allows for fast rule execution, Nisbet's approach allows for easier code generation and enables higher level of maintainability. In this research Nisbet's approach has been adopted because the main focus is to represent building code rules in a computable format independent of compliance checking systems. If system performance proves to be a serious issue for future compliance checking systems, such systems should be able to employ their own representations of the code that can be derived from a digital representation which is implementation neutral.

After the structure of the building code representation has been determined, the building code is modeled based on the developed representation model. As explained in section 3.4 the developed representation model consists of four levels:

1. Domain level
2. Rule level
3. Rule-set level
4. Management level

The representation stages cover respectively these levels as modeling steps.

3.5.2.1. Representation of Domain Concepts

The first step in the representation stage is the modeling of the building code domain concepts as object classes that form the domain level of the proposed representation model. For creating domain object classes, concepts in the building code document are identified and modeled with their attributes and their relationships to each other. In this step, building code domain experts and programming experts work together. This process is undertaken in three stages:

1. Extracting and listing the concepts and entities referenced in the building code document.
2. Identifying the required attributes of the objects and determining the relationships between them.
3. Implementing the representation as a library of objects in a computer-based form

The output of this step is a domain model, which will be utilized when modeling rules in the second step. For creating a domain model, two possible modeling approaches exist. The first involves modeling every concept of building code in a class hierarchy. In this approach specialized concepts are represented as sub-classes of the general concept classes. (e.g., a "kitchen door" can be modeled as a subclass of "door".) However, this approach tends to increase the complexity of the domain model. The second involves, instead of modeling every concept as a class, the creation of a concept-mapping table. This table lists all concepts of the building code and determines how each concept is represented; either as a class or a filtered set of instances within a class. (e.g., a "kitchen door" can map to all instances of "door" objects with the "relatedSpace" attribute value of "kitchen".) While the first approach can be useful when modeling simpler code documents, for modeling building codes that include a high number of concepts with complex relationships, the second approach should be adopted since a high number of specialized sub-classes can be avoided.

3.5.2.2. Representation of Rule Statements

The second step in the representation stage involves modeling individual rule statements of the building code as rule objects in the form of structured data. The rules are modeled using the rule model schema developed as part of the proposed representation model. In this step, rule statement semantics of the building code are captured in rule objects. These rule objects form the rule level of the proposed representation model. This process is carried out by building code domain experts and involves breaking down of the building code rule statements into its constructs and modeling rules by utilizing the concepts modeled in the lower domain level. The output of this step is a rule model covering separate rule objects indicating a single requirement of the building code rule statements.

3.5.2.3. Representation of Relationships between Rules

The third step in the representation stage of the building code modeling methodology involves defining relationships among rule objects modeled in the second step. In this step, related rule objects that are associated with the same subject are collected together into computable rule-sets by using logical conjunctions. These rule-sets form the rule-set level of the proposed representation model. This process is handled by building code domain experts and involves using two logical conjunctions "AND" and "OR" for connecting rule objects. AND conjunction is for connecting rule objects indicating the different conditions that apply simultaneously for the same subject. OR conjunction is for connecting rule objects that indicate alternative requirements for the same subject depending on specified conditions. The output of this step is the rule-set model covering a collection of rule-set objects. This rule-set model is the logical combination of distinct rule objects.

3.5.2.4. Representation of the Building Code Organization

The fourth step in the representation stage of the building code modeling methodology is the modeling of alternative organizations of the overall building code representation by categorizing rule-set objects modeled in the third step. 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 exist independently at this level. The output of this step is the rule-set classification objects, which will be employed by compliance checking algorithms to identify all needed rules to be processed for a given project. Building code domain experts carries out this step.

3.5.3. Implementation Stage

During the second stage, which is named as the representation stage in the proposed building code modeling methodology, the building code is modeled based on the new method of building code representation developed as part of this thesis. Implementation of the building code model within a compliance checking application is the third stage in the proposed building code modeling methodology for development of building code representations. This stage consist of two steps: 1) Implementation of the building code model and 2) testing and validation.

3.5.3.1. Implementation of the Building Code Model

It is important to actually implement the building code representation in compliance checking applications as part of the development process. Through the implementation process, several ambiguities, unclear points, missing definitions (concept, rule, relationship), and insufficient scope definitions can be revealed. Answers to questions such as, “Is all the required information for calculating a window opening area available in the building information model?” can only be validated through implementation. Moreover implementation is necessary for the purpose of demonstration and evaluation of the developed model for the representation of building codes. The developed representation model is intended to be utilized as a basis for compliance checking systems. Thus it should be demonstrated through an actual implementation.

One important issue during the implementation of the building code representation in a compliance checking application is related to the identification of building information modeling requirements for the building code domain. Building information models represent a building in digital format, hold all necessary information about the design, and are checked against the code for compliance. However, building information models created by a typical BIM platform such as REVIT or ARCHICAD, to date, do not include the level of detail needed for most building codes. For this reason, modeling requirements for building code domains should be identified. This information must then be used to extend the BIM standard (IFC) and built into the design software to allow proper data exchange. The domain model which is developed as the first level of the representation, in fact, embodies the modeling requirements for building code domains. How much of the required data can be obtained from the building information models should be analyzed in the implementation stage. This information can be used to inform efforts towards establishing BIM standards.

This stage is carried out by software developers and involves identifying modeling requirements by utilizing the domain objects modeled in the first step of the representation stage, and methods of mapping the domain model to the building information model. The output of this step is a running system that is able to execute compliance checking of building projects modeled in BIM environments against the building code representation.

3.5.3.2. Testing and Validation

The second step in the implementation stage is testing and validation of the building code model. The Building code model should be evaluated in terms of the three requisite properties found in literature (completeness, uniqueness, and correctness). These requisites are used to evaluate whether building code models are appropriately represented in computational format by most of the research on representation of building codes. These requisite properties are defined as follows (Gero, 1984; Fenves et al., 1987):

- Completeness, meaning that the code model can be applied to all possible situations (conditions) within its scope;
- Uniqueness, meaning that the model has no redundant rules and has no contradicting rules and generates the same unique result every time, when applied under a given set of conditions;
- Correctness (clarity), meaning that the result of applying the model must be consistent with the objective of the building code.

Fenves states that completeness and uniqueness are syntactic properties that are related to the organization of the code, while correctness is a semantic property that is more related to the meaning. The new building code representation is based on the RASE constructs and each rule statement is modeled individually. This makes guaranteeing completeness simple and is a major strength of the approach. By ensuring that all statements identified (to be within scope) in the analysis stage are modeled completeness can be guaranteed.

Uniqueness ensures that only one rule is applicable for any given situation. Uniqueness can also be defined as the lack of redundancy and lack of contradiction. A rule object is said to be redundant if its applicability conditions are guaranteed to be superseded by other rules. A set of rules is said to be in contradiction when they are all applicable for a given situation (condition). The four level structure of the new representation follows Nyman's proposed structure for building code representations.

The third level which is the rule-set level of the new building code representation is designed to expose the relationships between the rules. Each rule-set deals with a single property of a single domain object and all rules related to the property are collected under a single tree based on the applicability conditions of each rule. Only one rule from the tree is selected as applicable and thus contradictions are not possible. The explicit modeling of conditions in the rule-set tree makes it simple to ensure that there exists a set of conditions for selecting each rule and thus redundancies are avoided. These features of the new representation ensures uniqueness.

Correctness ensures that rule objects represent the meaning, intentions, and implications of the corresponding rule sentences correctly. Completeness and uniqueness are syntactic properties and the representation can guarantee them, but correctness is semantic. The developed building code model should be tested for correctness. The results need to checked and validated by building code domain experts preferably by ones outside the core committee developing the building code representation. Validation should be done using the running compliance checking system built in the first step. The testing of the building code model should be carried out using specially prepared test cases. The test cases should include both valid and invalid building design instances for each and every rule in the code. The output of this step in the building code modeling process is a human readable document validating the building code representation. If the results of the testing identify erroneous or incomplete modeling, this information is used as input to start another iteration of the representation stage in order to resolve the issues. If there are no errors, the modeling process ends.

CHAPTER 4

IMPLEMENTATION AND EVALUATION

In order to provide a proof-of-concept implementation for the newly developed representation model, a case study has been conducted. The case study focused on modeling an actual building code and illustrating the use of this model within future compliance checking applications. 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. From IMHZCode, the subset of all clauses pertinent to buildings has been modeled. This implementation illustrates the process for representing an existing building code.

The case study has been carried out in 3 stages following the proposed building code representation methodology in section 3.5. Next sections explain these steps in detail and show the results.

Stage 1: Analysis

- Scope
- Decomposition
- Classification

Stage 2: Representation

- Domain objects
- Rule objects
- Rule-set objects
- Rule-set group objects

Stage 3: Implementation

- Prototype
- Testing and Validation

4.1. Analysis

4.1.1. Scope

In order for the new model to be applicable to as wide a range of code documents as possible, the case study needed to focus on a complex building code with a large set of rules. To determine which building code will be modeled in the case study, current building codes in Turkish Architecture, Engineering and Construction (AEC) industry have been examined. In Turkey, every building project is checked against primarily the housing and zoning code of municipality where the building will be built. The municipalities' housing and zoning codes include rules defined by the ministry documents and add further specifications. Being based on the ministry documents, all housing and zoning codes contain similar rules with few exceptions. Building codes get tested primarily in municipalities of large cities where unforeseen cases and situations come up and force clarifications of code. İzmir is the third most populous city in Turkey and its housing and zoning code is representative of codes that are in effect throughout Turkey. In this respect, İzmir Municipality Housing and Zoning Code (IMHZCode) has been chosen for the case study.

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 divided into six parts as illustrated in Table 4.1.

Table 4.1. The Structure of IMHZCode

Part Id	**Part Heading**	**Clauses**
I	General Rules	1 - 10
II	Definitions	11 - 23
III	Rules Related to Buildings and Land Readjustment	24-66
IV	Rules Related to Construction Permit and Building Occupancy Permit	67-76
V	Buildings, Building Parts and Facilities Subject to Special Rules	77-86
VI	Rules Rescinded, Interim Provisions and Entry in to Force	87-89

The rules related to buildings are covered by the clauses that are included in part III whereas the rest of the building code is either informative or unrelated to buildings. Part III also includes clauses related to other subjects. Table 4.2 shows all clauses of Part III and the issues of their relevance. Each of the clauses pertinent to buildings consists of several rules defining constraints relating to specific concepts such as roofs, windows, doors, staircases etc. For the case study, IMHZCode's clauses that include rules pertinent to buildings are modeled based on the developed representation model.

In the next stages of the case study IMHZCode is analyzed in detail. First, the clauses related to buildings are decomposed into a list of statements. Then all statement types that exist are determined. After that, rule statements are classified according to their self-containedness and formalizability in order to document how much of the code as well as which types of rules can be modeled.

4.1.2. Decomposition

In this stage, IMHZCode's clauses that include rules pertinent to buildings are extracted. 27 clauses are found on buildings and these clauses are decomposed into a list of statements. As a result of the decomposition study, the statement list containing all 297 individual statements that form the clauses related to buildings is obtained. Afterwards, the type of each statement is determined as being one of "clarification", "applicability condition", or "rule". As explained in section 2.1.2, clauses include three types of statements: Clarifications, applicability conditions, and rules. Since only rule statements will be represented in computer implementable format, they are identified and extracted. 258 rule statements are found. The decomposition of all 27 clauses is given in Appendix A. The decomposition of Clause-27 and Clause-47 are given below in Table 4.3 and Table 4.4 as an example.

Table 4.2. The clauses of Part III of IMHZCode

Clause Id	**Clause Heading**	**Pertinent to ….**
24	Width of Parcels	Land Readjustment
25	Layout of parcels	Land Readjustment
26	Arrangement of parcels	Land Readjustment
27	Distance of Setbacks	Building and Land Readjustment
28	Depth of Buildings	Building
29	Façade of Buildings	Building
30	Height of Buildings	Building
31	Temporary Constructions	Construction Permit issue
32	Closed Roads and Streets	Land Readjustment
33	Flood Areas	Land Readjustment
34	Non-resettlement Areas	Land Readjustment
35	Construction Permits in Cadastral Parcels	Construction Permit Issue
36	Multiple Construction Permits in a Parcel	Construction Permit issue
37	Issuance of Ground Level	Land Readjustment
38	Slab Levels of Ground Floor	Building
39	Requirements in Some Buildings	Building
40	Eaves and Sun-shadings	Building
41	Roofs	Building
42	Cantilevers	Building
43	Light-shafts and Air-shafts	Building
44	Spaces and Dimensions	Building
45	Interior Heights	Building
46	Windows	Building
47	Doors	Building
48	Lifts	Building
49	Stairs	Building
50	Fire Escapes	Building
51	Balustrades	Building
52	Chimneys	Building
53	Fire Precautions	Building
54	Water Tank And Sanitary facilities	Installation Issue
55	Provisions Pertinent to Basement	Building
56	Porter Suite	Building
57	Auxiliary Buildings	Building
58	Lightning Rods, Central TV Antennas, AC	Installation Issue
59	Walls	Building
60	Fences	Building
61	Buildings of Construction Site	Construction Issue
61	Cesspools	Installation Issue
63	Arcades	Landscape Issue
64	Garden Arrangements & Building Aesthetics	Landscape Issue
65	Shelter	Building
66	Provisions for Disabilities	Building

Table 4.3. Decomposition of IMHZCode Clause-27

	Textual Expressions of A Clause	Statement Type
	Clause 27– Setback Distances *Id&heading*	
1	In cases where setback distances are not determined by the zoning plan in effect, setback distances have to be determined according to the conditions below.	Applicability Condition
	A- Front Setbacks: *subheading*	
2	Setbacks where there is a front yard and setbacks from roads, green areas and parking lots are at least 5.00 m.	Rule
3	However, on blocks that have existing buildings (except detached order blocks) setbacks will be determined according to the following conditions taking into account existing buildings on the same block facade.	Applicability Condition
4	a) In Semi-Detached Building Blocks, If there is an existing building in one of the two lots then setbacks will be determined based on the existing building.	Rule
5	b) In Planned Unit Developments, If any of the lots have an existing building then only for this block, setbacks for the lots are based on setbacks of the existing building.	Rule
6	c) In Attached Building Blocks, If more than 50% of the block façade has been developed within the height limits of the zoning plan then setbacks are determined based on the existing buildings with the same height.	Rule
	B-Side Setbacks: *subheading*	
7	Side setbacks (up to and including 4 storeys) shall be 3.00 meters.	Rule
8	For buildings taller than this side setbacks are increased by 0.5 meters for every additional storey.	Rule
9	However, for timber-framed buildings side setbacks must be at least 5.00 meters.	Rule
	C- Rear Setbacks: *subheading*	
10	Rear setbacks are H/2.	Rule
11	H is the height of building and is determined according to clause 30 of this code.	Clarification
12	Rear setbacks also apply to lots with single road frontage, 2 road frontages (corner lots) and corner lots with 3 road frontages.	Clarification
13	On blocks that have existing buildings setbacks will be determined according to the following conditions taking into account existing buildings on the same block facade with the condition that rear setbacks will never be less than 3.00 m.	Rule & Applicability Condition
14	a) In Semi-Detached Building Blocks, If there is an existing building in one of the two lots then rear setbacks will be determined based on the existing building.	Rule
15	b) In Planned Unit Developments, If any of the lots have an existing building then only for this block, rear setbacks for the lots are based on setbacks of the existing building.	Rule
16	c) In Attached Building Blocks, If more than 50% of the block façade has been developed within the height limits of the zoning plan then setbacks are determined based on the existing buildings with the same height.	Rule

Table 4.4. Decomposition of IMHZCode Clause-47

Textual Expressions of A Clause		Statement Type
Clause 47– Doors	*Id&heading*	
1	Clear height of doors shall be at least 2.10 m	Rule
Width of Doors:	*subheading*	
2	Clear width of main entrance doors of buildings, which has multiple independent unit, shall be at least 1.30 meters.	Rule
3	Clear width of entrance doors of independent unit shall be at least 1.00 meters	Rule
4	Clear width of room and kitchen doors shall be at least 0.90 meters.	Rule
5	Clear width of bathroom, WC, cellar doors shall be at least 0.80 meters.	Rule
6	Clear width of store doors shall be at least 1.00 meters.	Rule
7	Dimensions of garage, elevator, and similar technical spaces doors shall be determined in the manner required by the service.	Rule
8	The bathroom doors must allow air transfer from the bottom part.	Rule

4.1.3. Rule Classification

IMHZCode is a written text document and it has a complex structure. Classification of rules based on the code structure is needed for understanding higher-order relationships between rule statements. The analysis of IMHZCode structure has revealed two types of rules:

- Self-contained rules
- Linked explanatory rules

Self-contained rules indicate how something will be, must be, should be, or can be. The rules related to the width of the stairs, the setback distances, and the height of entrance doors are examples of this type:

"The minimum width for flights and landings of stairs shall be 1.20 meters."
"Side setbacks (up to and including 4 storeys) shall be 3.00 meters."

"Clear height of doors shall be at least 2.10 m"

Linked-explanatory 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."

Another example from IMHZCode modifies the rule on the minimum distance for side setbacks.

"For buildings taller than this (4-storeys) side setbacks are increased by 0.5 meters for every additional storey."

In addition to a complex structure, IMHZCode 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 the IMHZCode representation. Three additional types of rules have been identified:

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

Formalizable rules are straightforward and can be clearly represented in a computer implementable format. They can be modeled in a single step by the selected representation method. These types of rules allow for automated compliance checking without any ambiguities. The followings are examples from IMHZCode:

"Clear width of room and kitchen doors shall be at least 0.90 meters."

" Roof slope cannot exceed 33%."

Semi-formalizable rules contain ambiguous or fuzzy concepts that require human 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. The required clarifications of concepts are possible by employing objective metrics such as minimum or maximum distances. Example:

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

Non-formalizable rules 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."

IMHZCode rule statements on buildings are classified based on the types of rules that have been identified through this analysis. The two classifications have been presented in this section. One classification has been based on the structure of the document. Rules are either self-contained or linked to other rules providing further explanations. The second classification is based on the rules' formalizability. Some rule statements can be represented in computational form, some require human interpretation but can be computationally supported if appropriate objective measures are employed during checking while others clearly cannot be subject to automated reasoning. The classification of the rule statements of Clause-27 and Clause-47 are given in Table 4.5 and Table 4.6 as an example. The classification of all 258 rule statements is provided in Appendix A.

Table 4.5. Classification of the rule statements of Clause-27

Clause Id	**Statement Id**	**Rule Id**	**Rule Type**	
			Self-containedness	**Formalizability**
C27	ST27.2	R27.1	Self-contained	Formalizable
	ST27.4	R27.2	Linked-explanatory	Formalizable
	ST27.5	R27.3	Linked-explanatory	Formalizable
	ST27.6	R27.4	Linked-explanatory	Formalizable
	ST27.7	R27.5	Self-contained	Formalizable
	ST27.8	R27.6	Linked-explanatory	Formalizable
	ST27.9	R27.7	Linked-explanatory	Formalizable
	ST27.10	R27.8	Self-contained	Formalizable
	ST27.13	R27.9	Linked-explanatory	Formalizable
	ST27.14	R27.10	Linked-explanatory	Formalizable
	ST27.15	R27.11	Linked-explanatory	Formalizable
	ST27.16	R27.12	Linked-explanatory	Formalizable

Table 4.6. Classification of the rule statements of Clause-47

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	**Formalizability**
C27	ST1	R47.1	Self-contained	Formalizable
	ST2	R47.2	Linked-explanatory	Formalizable
	ST3	R47.3	Linked-explanatory	Formalizable
	ST4	R47.4	Linked-explanatory	Formalizable
	ST5	R47.5	Linked-explanatory	Formalizable
	ST6	R47.6	Linked-explanatory	Formalizable
	ST7	R47.7	Linked-explanatory	Non-formalizable
	ST8	R47.8	Self-contained	Formalizable

The classification study revealed that 58% of the 258 rules that are found are self-contained and formalizable and 21% are explicative and formalizable. As indicated in Table 4.7, 79% of IMHZcode rules on residential buildings can be represented in computer implementable format.

Table 4.7. Results of the classification of IMHZCode rule statements on buildings

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

In this case study, all formalizable rules of IMHZCode are studied for the representation in a computer implementable format by using the developed representation model. The semi-formalizable rules, which are based on fuzzy concepts that introduce ambiguities though they can be clarified, have been left out of the case study after a brief investigation into this class of rules. Handling these rules require the input of code authors as well as authorities in charge of code compliance checking. Options in objectifying various criteria need to be explored, however it is not possible to identify options without a proof-of-concept system. Studying the strategies into how best to deal with semi-formalizable rules can only be conducted after a robust methodology for representing formalizable rules have been established. The existence of non-formalizable rules in building codes is a separate research topic and is not within the scope of this research.

4.2. Representation

During the analysis stage, the building code, the chapter of the code, and the clauses of the chapter to be represented are determined and all rules in selected clauses are classified. Afterwards, the modeling of IMHZCode's formalizable rules on buildings was carried out. IMHZCode's all formalizable rule statements on buildings have been modeled based on the new method of rule representation developed as part of this thesis. Next sections explain the modeling steps in detail.

4.2.1. Domain Objects and Concept Mapping List

As explained in section 3.4.1 a computer representation for building codes needs to build upon domain objects that reside in the lowest level of the four level representation. 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. For example, "construction technique" is a term that is mentioned in IMHZCode and it is an attribute of the "building" concept. There are also terms like height, width, etc. which are mostly used to define requirements. After all terms are 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. When modeling rules, domain classes are used for building applicability and selection constructs. The UML diagrams for the resulting IMHZCode domain model is given in Figure 4.1 and Figure 4.2.

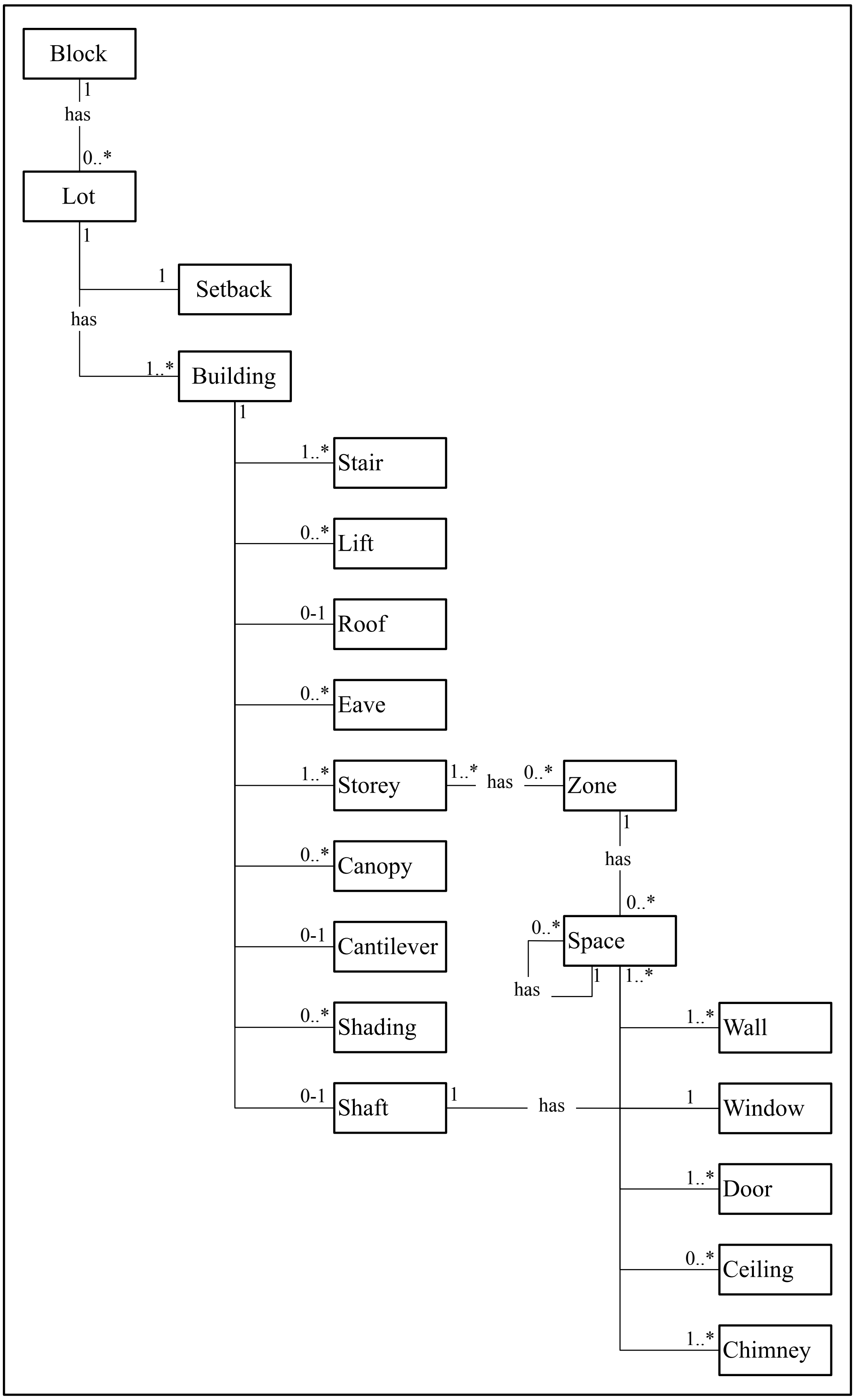


Figure 4.1. Domain objects of IMHZCode

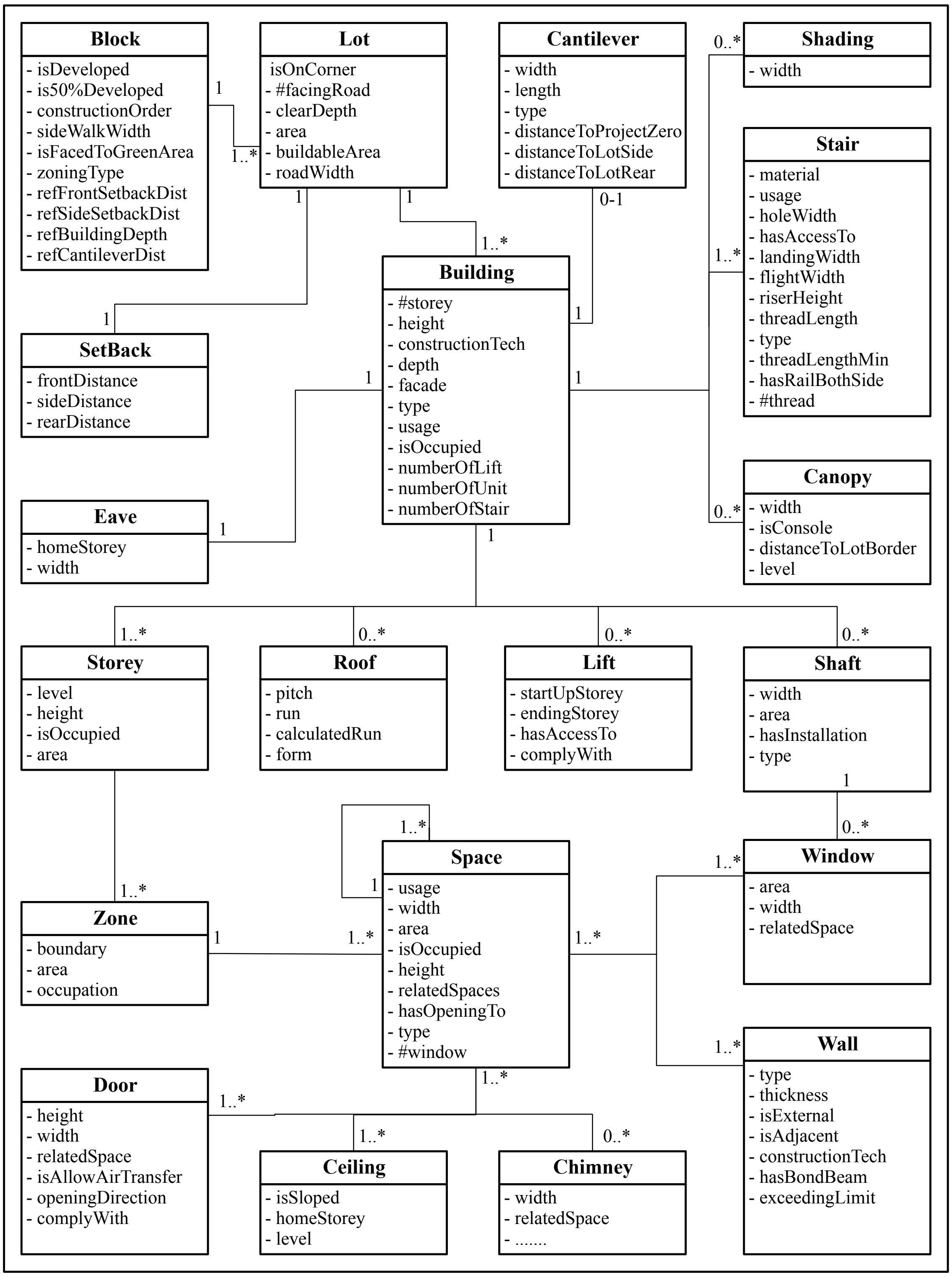


Figure 4.2. The domain model of IMHZCode

While concepts generally correspond to a class in the domain model, some concepts correspond to a subset of objects that belong to a class. All objects in the subset are required to hold a specified value for a certain property. For example, a "door" concept is directly represented as a *Door* object with attributes such as *height*, *width*, *relatedSpace*, *allowAirTransfer*. A "bathroom door", on the other hand is a *Door* with "bathroom" as its *relatedSpace*. These specialized concepts can naturally be modeled through inheritance following the normal object oriented paradigm. However, extending the class hierarchy only for selecting specific subsets may quickly and needlessly increase the complexity of the domain model. All possible variations of doors ("bathroom door", "kitchen door", "entrance door", etc.) should not need to force the modeling of individual classes.

The proposed model includes a mapping list for such concepts that are required for the selection of subsets. Instead of modeling every concept as a class, a concept-mapping table is created that defines how a concept maps to a filtered set of objects. The concept mapping table for IMHZCode is shown in Table 4.8. By associating concepts with a set of selection criteria, the need for a high number of specialized sub-classes derived from the main domain classes was eliminated.

Table 4.8. Concept-mapping table for IMHZCode

	SELECTION FILTER			
Concept	**Class**	**Property**	**Comparator**	**Value**
"attic"	Zone	occupation	equal	attic
"independentUnit"	Zone	occupation	equal	independent unit
"liftShaft"	Shaft	type	equal	lift shaft
"airShaft"	Shaft	type	equal	air shaft
"coalCellar"	Space	usage	equal	coal cellar
"roofTerrace"	Zone	occupation	equal	roof terrace
"stairwell"	Shaft	type	equal	stair shaft
"gableWall"	Wall	type	equal	gable
"mainEntranceDoor	Door	relatedSpace	equal	mainEntrance
"bathroomDoor"	Door	relatedSpace	equal	bathroom
"kitchenDoor"	Door	relatedSpace	equal	kitchen
"entranceDoor"	Door	relatedSpace	equal	entrance
"roomDoor"	Door	relatedSpace	equal	room
"cellarDoor"	Door	relatedSpace	equal	cellar
"livingRoom"	Space	usage	equal	livingRoom
"kitchen"	Space	usage	equal	livingRoom
"bedroom"	Space	usage	equal	livingRoom
"bathroom"	Space	usage	equal	livingRoom
"basement"	Storey	level	equal	basement
"dwelling"	Zone	occupation	equal	dwellingUnit

The concept-mapping table makes concepts available as an optional list for specifying the *selection* construct of a rule object in the level above. This mapping table acts as a library of a pre-determined *selection* constructs for rule objects with concepts as key (or reference). When modeling rules in the level above, a separate selection construct need not be modeled in each rule that refers to the same concept. All rules can refer to the concept and the selection construct can be picked up from the mapping table. This ensures consistency in the model and prevents unnecessary repetitions. When a selection construct related with a concept needs to change, only the mapping table needs to be updated for that concept.

4.2.2. Rule Objects

Individual rule statements in 27 clauses of the IMHZCode have been 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.

As explained in the section 3.4.2 the *concept* is a description of the subject to which the rule applies and the *property* is an attribute of the *concept*. The concepts and their properties are defined in the domain model. The requirement constructs must refer to concepts modeled as classes in the domain model. The selection constructs on the other hand may additionally make use of the specialized concepts in the concept mapping table.

The *comparator* is a numeric comparison operator such as "≥", "≤", "=", if the value is numeric. If the value is Boolean, then only the "boolean" comparator is used. If the value is descriptive, then the "equal" or "!equal" comparators are used. If the value represents a set of concepts, then the comparator is any of the set comparison operators such as "includes", excludes".

The *value* is the specific value that is found in the code, whether numeric, descriptive, or Boolean. There are two different kinds of values: Explicit (literal value) and derived (an expression). While explicit value is a constant, derived value is either a reference to another concept's property or the result of a mathematical expression.

Curly brackets “{}” are used for specifying references to concepts, and parentheses “()” are used to specify expressions.

The *unit* specifies the unit of measure for numeric values. If the value is not numeric the unit is blank.

The rule models of Clause-27 and Clause-47 are given in Table 4.9 and Table 4.10 as an example for illustrating how rule statements are modeled. The rule model of all 27 clauses is provided in Appendix B.

Table 4.9. 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_referencedFront SetbackDistance}	m	Block	constructionOrder	equal	semiDetached
						Block	hasExistingBuilding	boolean	true
R27.3	Setback	front Distance	=	{Block_referencedFront SetbackDistance}	m	Block	constructionOrder	equal	plannedUnit
						Block	hasExistingBuilding	equal	true
R27.4	Setback	front Distance	=	{Block_referencedFront SetbackDistance}	m	Block	constructionOrder	equal	attached
						Block	is50%Developed	equal	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_referencedRear SetbackDistance}	m	Block	constructionOrder	equal	semiDetached
						Block	hasExistingBuilding	boolean	true
R27.11	Setback	rear Distance	=	{Block_referencedRear SetbackDistance}	m	Block	constructionOrder	equal	plannedUnit
						Block	hasExistingBuilding	boolean	true
R27.12	Setback	rear Distance	=	{Block_referencedRear SetbackDistance}	m	Block	constructionOrder	equal	attached
						Block	is50%Developed	boolean	true

Table 4.10. The structured rule objects of IMHZCode Clause-47

Rule	REQUIREMENT					SELECTION			
Id	Concept	Property	Comp.	Value	U.	Concept	Property	Comparator	Value
R47.1	Door	height	≥	2.10	m				
R47.2	Door	width	≥	1.30	m	mainEntranceDoor			
R47.3	Door	width	≥	1.00	m	entranceDoor			
R47.4.1	Door	width	≥	0.90	m	roomDoor			
R47.4.2	Door	width	≥	0.90	m	kitchenDoor			
R47.5.1	Door	width	≥	0.80	m	bathroomDoor			
R47.5.2	Door	width	≥	0.80	m	wcDoor			
R47.5.3	Door	width	≥	0.80	m	cellarDoor			
R47.6	Door	width	≥	1.00	m	storeDoor			
R47.8	Door	allowAirTransfer	boolean	true		bathroomDoor			

In the code document, individual rule statements generally indicate a single requirement, which has a single subject and a single predicate, associated with a concept. However, some individual rule statements of IMHZCode indicate multiple requirements of a concept or a requirement related to multiple concepts. For example, the following individual rule statement indicates a requirement about two concepts, room and kitchen doors:

> "Clause 47 – Doors:
> … Clear width of room and kitchen doors shall be at least 0.90 meters.… :"

> "Madde 47 – Kapılar:
> … Oda ve mutfak kapıları kasa dahil (0.90) metreden az olamaz…:"

For another example, the following individual rule statement indicates two requirements about a single concept, one qualifies the width and the other qualifies the area of light shafts:

> "Clause 43 – Light and air shafts:
> … The width of light shafts shall not be less than 1.00 m and the area of them shall not be less than 3.00 m2 in one or two-storey buildings … :"

> "Madde 43 – Işıklıklar ve hava bacaları:
> … Işıklıklar, 1 ve 2 katlı binalarda; dar kenarı 1.00 metreden, alanı 3.00 m2'den az olamaz …:"

When modeling, these types of rule statements need to be separated into multiple statements, each targeting a single requirement related to the same concept. Each rule object thus indicates a single requirement and is associated with a single property of a single concept defined in the domain model. The relationships among these individual rule statements are modeled in the levels above.

4.2.3. Rule-set Objects

IMHZCode is 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. The rule-set model defines the relationships among individual rule statements as explained in section 3.4.3. Related rule objects that are associated with the same property of the same concept are collected together and modeled as nested rule-sets using 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. The rule-set objects of Clause-27 and Clause-47 are given in Table 4.11 and Table 4.12 as an example. The collection of rule-set objects of IMHZCode is provided in appendix X

Table 4.11. 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)))

Table 4.12. The rule-set objects of IMHZCode Clause-47

Id	**Subject**		**Set**
	Concept	**Property**	
RS47.A	Door	height	R47.1
RS47.B	Door	width	(||: R47.2, R47.3, R47.4.1, R47.4.2, R47.5.1, R47.5.2, R47.5.3, R47.6)
RS47.C	Door	allowAirTransfer	R47.8

Some rule sets have a simple, one level relation between rule objects. The rule-set RS27.A is the collection of rules specifying constraints for the *frontDistance* property of the *Setback* concept and is an example for this type of rule-sets. (Figure 4.3)

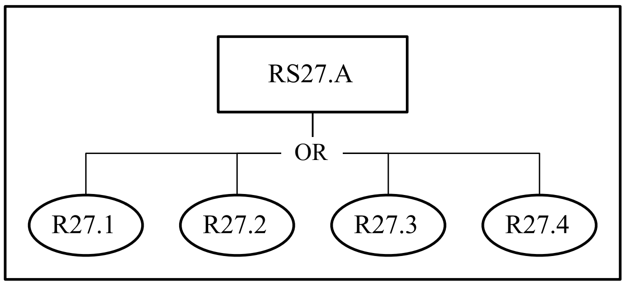


Figure 4.3. Tree representation of the rule-set RS27.A

Some rule sets have multilevel relations between rule objects. The nested sets of rules form a hierarchical tree-structure. Rule-sets RS27.B, and RS27.C are collections of rules specifying *sideDistance*, and *rearDistance* properties of the *Setback* concept, and are examples for this type of complex rule-sets. (Figure 4.4)

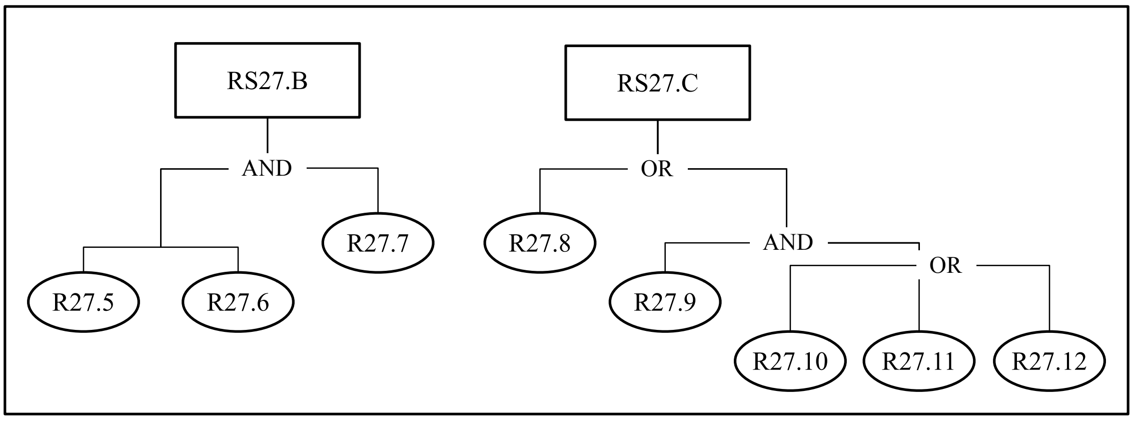


Figure 4.4. Tree representation of rule-set RS27.B and RS27.C

4.2.4. Rule-set Group Objects

The final fourth level (*management level*) of the new 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. There may be multiple methods of grouping rule-sets each representing a different sorting scheme. One obvious grouping method is the structure of the code document itself. Rule-sets can be grouped representing the heading and sub-heading based structure of the document. Many other types of relationships and similarities also exist among rules and rule-sets that can be used as criteria for grouping them. Alternative methods of grouping rule-sets are allowed to exist. The case study on the IMHZCode has shown that grouping rule-sets based on the concept (corresponds to a class in the domain model) they are related to is preferred by compliance checking algorithm employed by the prototype implementation. 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 before moving on to other classes of objects. The prototype implementation is described in the next section. Table 4.13 illustrates classification of rule-sets of *setback, building,* and *door* concepts as an example.

Table 4.13. Classification of rule-sets related to setback, building, and door 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
	Door		
		height – [RS47.A=(R47.1)]	
			R47.1
		width – [RS47.B = (‖: R47.2, R47.3, R47.4.1, R47.4.2, R47.5.1, R47.5.2, R47.5.3, R47.6)]	
			R47.2
			R47.3
			R47.4.1
			R47.4.2
			R47.5.1
			R47.5.2
			R47.5.3
			R47.6
		allowAirTransfer – [RS47.C = (R47.8)]	
			R47.8

4.3. Implementation

During the modeling stage, IMHZCode's all formalizable rule statements on buildings have been modeled based on the new method of rule representation developed as part of this thesis. Afterwards, the implementation of IMHZCode model and its utilization within a compliance checking application was carried out. Currently, two commonly used compliance-checking systems 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 was found suitable as a test bed for the IMHZCode model. For this reason a new code compliance checking system has been implemented as a prototype for testing even though the scope of this research does not cover developing automated compliance checking systems. This proof-of-concept prototype demonstrates the feasibility of the proposed model.

4.3.1. Prototype

The prototype has been developed solely for the purpose of demonstration and evaluation of the proposed model for the representation of building codes. The prototype system has been implemented using the Java language and consists of three main components: Building Code Reader, Model View Builder, and Compliance Checker. Building Code Reader reads the building code model from the database and instantiates rule objects for checking, Model View Builder extracts objects and properties of interest from BIM data, populates the domain model objects and thus derives a model view to be checked. Compliance Checker ensures that the building project meets all requirements by applying all rules to related domain concepts and compiles a report. Figure 4.5 illustrates the conceptual framework for the compliance checking system.

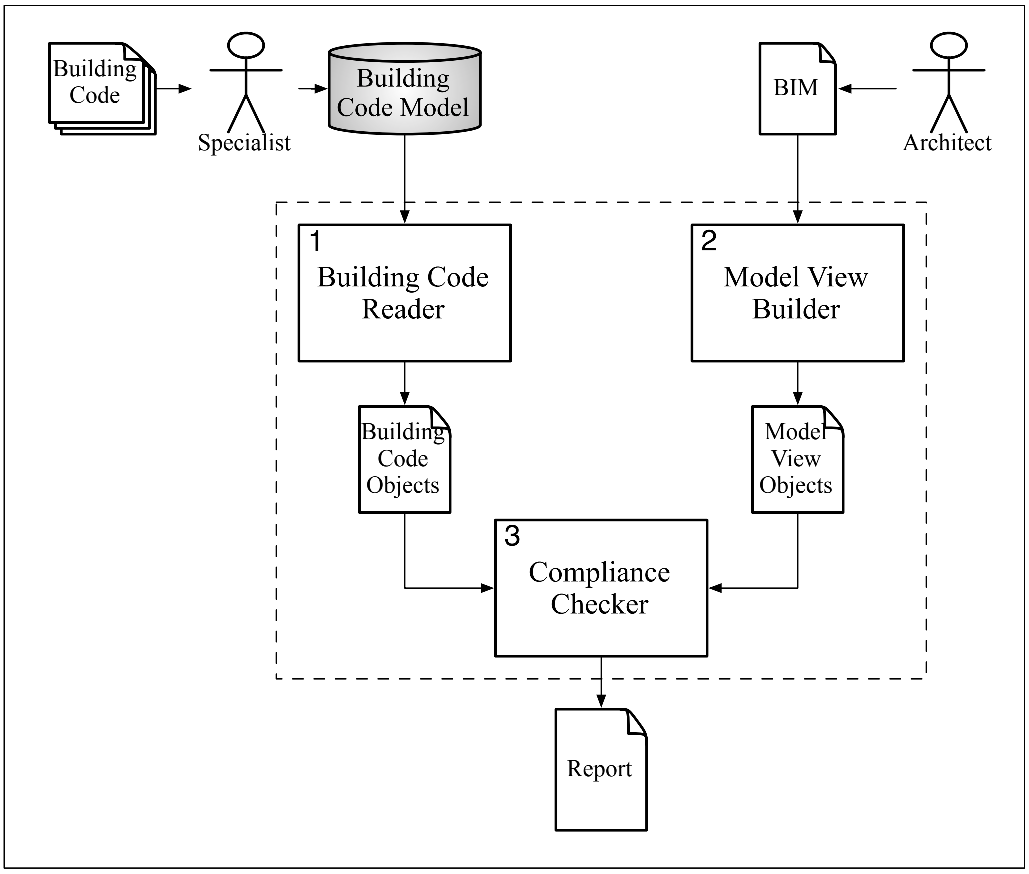


Figure 4.5. Conceptual framework for the compliance checking system

Future code checking systems can develop user-friendly interfaces for describing actual building codes in computational format which is developed through this research. In this prototype, the building code model is stored in a database. A relational database application with a graphical user interface (FileMaker Pro) is used as a tool to create and store the code model in computational format. Figure 4.6 illustrates tables and relations in the IMHZCode model database. 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. Exchange of the building code can happen through various methods. Code checking systems may establish live connections and retrieve latest codes from the appropriate authority. The code can be in a number of formats that can represent object oriented data, such as XML.

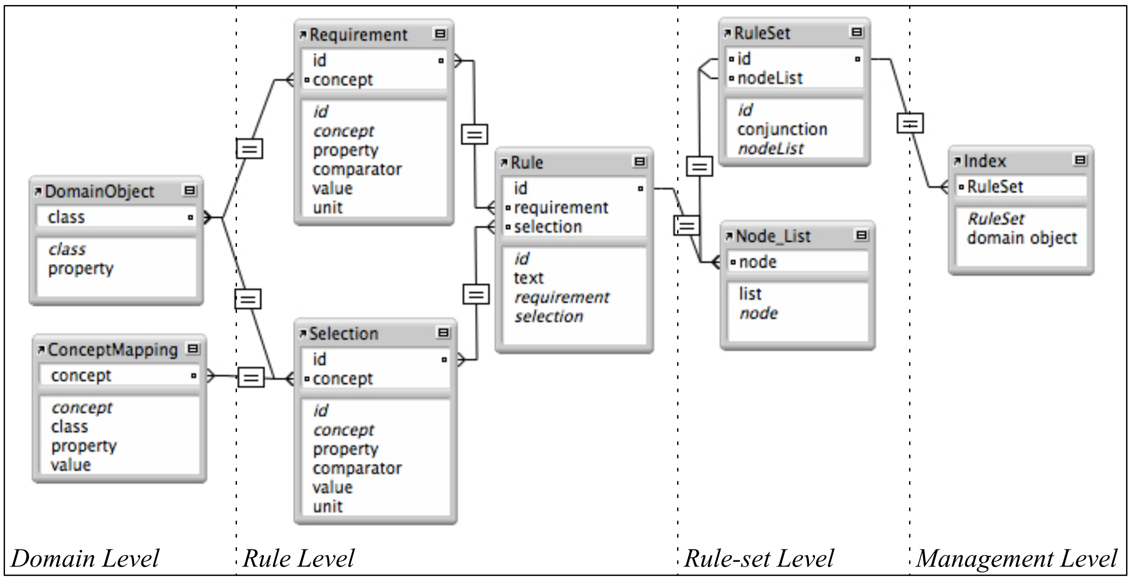


Figure 4.6. Database structure for the IMHZCode model

In the prototype, the Building Code Reader component connects to the database where the building code models are stored, reads from it and instantiates the necessary objects (rules, rule-sets, rule-set groups, etc.). Connection of the Building Code Reader to the IMHZCode database takes place via a JDBC-ODBC bridge that enables Java applications to query relational databases.

While rules are stored in a database, the project data to be checked for its compliance with the code is in the form of a building information model (BIM) file. In this prototype, the BIM model is required to be an Industry Foundation Classes (IFC2x3) file. The IFC format is utilized by most major research efforts in compliance checking. IFC is currently considered to be the most suitable schema for improving information exchange and interoperability in the construction industry.

Model View Builder component of the prototype accesses and extracts the BIM data that is required during compliance checking. The prototype makes use of JSDAI (Java Standard Data Access Interface) application programming interface for parsing STEP (Standard for the Exchange of Product Model Data – ISO 10303) files. IFC2x3 files exported from the BIM application are parsed using the JSDAI library and IFC objects are created. The IFC objects and their properties of interest are mapped to domain objects and information is copied over to the domain objects and thus the required model view is derived for the Compliance Checker.

The prototype's the third major component is the Compliance Checker. It applies rules to the derived model view and returns a report. Compliance Checker makes use of the classification of rule-sets by concept that was discussed in section 3.4.4 and section 4.2.4. Compliance Checker takes the list of "rule-set group objects" that group rule-sets according to the concept they constrain. For each group it applies all rule-sets to all instances of the concept found in the model view. An instance of a concept passes a check either if it is as required for every applicable rule in AND rule-sets or if it is as required for any applicable rule in the OR rule-sets. Finally, the Compliance Checker reports compliant and non-compliant instances of concepts and related rules. The functional diagram of this component is illustrated in Figure 4.7.

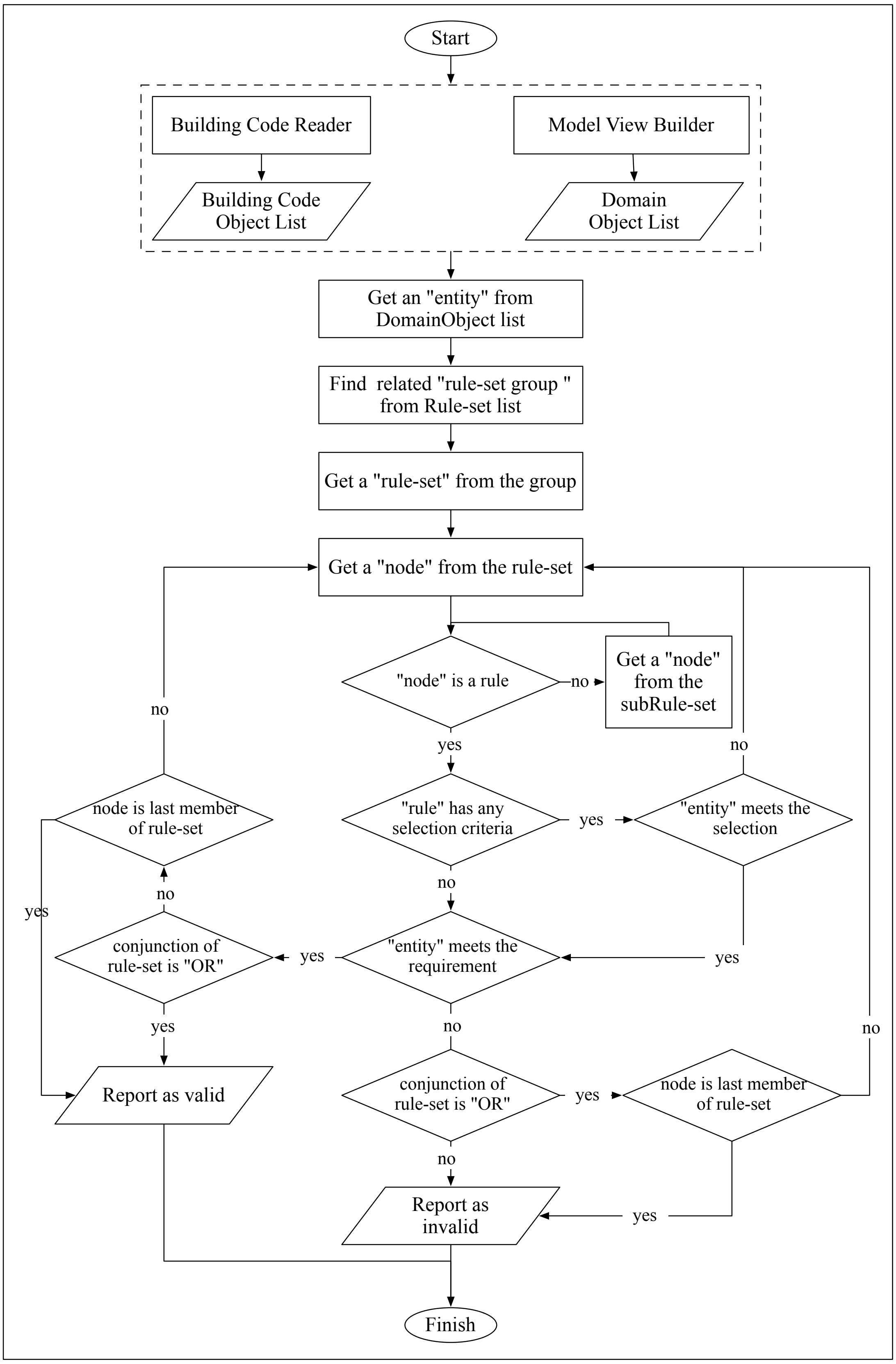

Start
Building Code Reader
Model View Builder
Building Code Object List
Domain Object List
Get an "entity" from DomainObject list
Find related "rule-set group " from Rule-set list
Get a "rule-set" from the group
Get a "node" from the rule-set
"node" is a rule
no
Get a "node" from the subRule-set
yes
no
no
"rule" has any selection criteria
yes
"entity" meets the selection
node is last member of rule-set
no
yes
no
conjunction of rule-set is "OR"
yes
"entity" meets the requirement
yes
yes
no
no
Report as valid
conjunction of rule-set is "OR"
yes
node is last member of rule-set
no
Report as invalid
yes
Finish

Figure 4.7. Functional diagram of Checker

4.3.2. Building Information Modeling Requirements

One important prerequisite for the development of working automated compliance checking systems is building information models that hold all necessary information about a building and its site for all codes which compliance will be tested. Automated compliance checking of a building project against a given building code, requires additional information from BIM concerning the specialized domain to which the code belongs to. Since the building information models created by BIM systems do not include the level of detail needed for IMHZCode, as explained in section 3.5.3, modeling requirements of the code should be identified.

The modeling requirements for the IMHZCode domain have been identified in order to develop the prototype system. Domain model for IMHZCode was developed as the first level of the new representation model. The IMHZCode domain model established the required objects and level of detail for building models. How much of the required data can be obtained from the BIM model was analyzed and methods of mapping the domain model to the BIM model are identified.

The analysis of the domain model to the BIM mapping process reveals that four types of mappings exist. The first type of mapping takes place between domain objects and their properties that can match directly to already existing BIM objects. As an example door object of the domain model and its properties such as width and height are available as *ifcdoor* objects. However, some required properties are missing in BIM objects. The second mapping type takes place when these missing properties can be derived from ifc relationship definitions between entities if it is possible. As an example relatedSpace property of door object can be derived examining the relationship between ifcdoor and ifcspace entities. The third mapping type takes place when deriving the required data via relationships in ifc is not possible. In these instances, missing properties are defined as additional IFC property sets for compliance checking. buildingSMART provides a methodology for defining additional properties (buildingSMART, 2008b). IFC allows extending the schema and adding custom properties by defining property sets (PSET). The PSETs and properties for applying the IMHZCode are shown in Figure 4.8. The fourth type of mapping takes place when domain concepts have no corresponding objects in the BIM model. These missing entities require updating the IFC schema in its upcoming versions. The concept of setbacks is one example. In current BIM representations there are no setback objects. Therefore, only for the prototype system, setback information has been defined as an extended property sets of ifcSite and the required data for setback is derived from that.

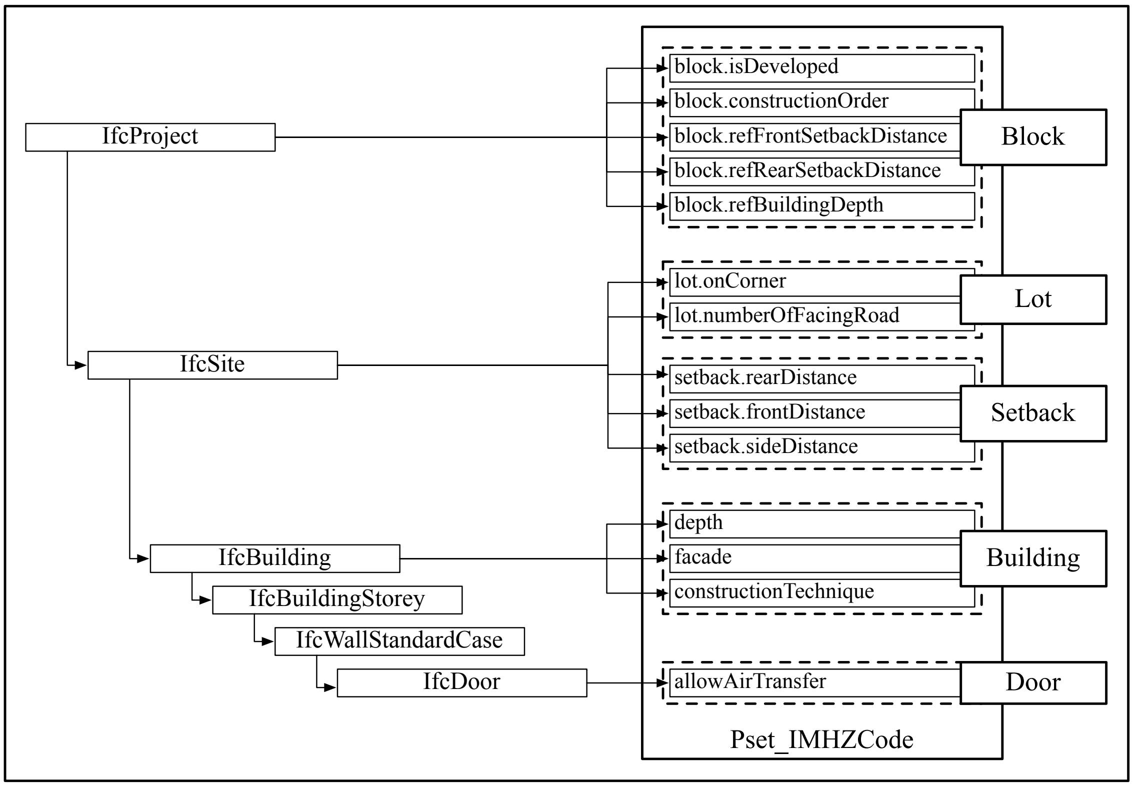


Figure 4.8. Definition of properties for applying the IMHZCode

4.4. Validation

The validation of the new representation model is carried out in three steps:

1. Representing an actual building code based on the developed model.
2. Implementing the code model in a compliance checking system.
3. Evaluating the code model in terms of correctness.

For the validation of the new representation model explained in the CHAPTER 3, a case study has been conducted. For the case study, İzmir Municipality Housing and Zoning Code (IMHZCode), which is representative of codes that are in effect throughout Turkey, has been chosen. IMHZCode is a complex building code that consists of various types of rule statements with dependencies. The 27 clauses related to buildings have been analyzed and modeled using the newly developed representation. The case study demonstrates that the developed representation model is capable of representing building codes in a computer implementable format.

The implementation of the IMHZCode model in a new compliance checking system developed as a prototype has been explained in section 4.3.1. The implementation has been tested on a range of different building projects exported by a BIM tool.

Correctness ensures that rule objects represent the meaning, intentions, and implications of the corresponding rule sentences correctly. The correctness of the IMHZCode model has been evaluated by using the prototype to carry out testing and validation. Simple building models have been prepared in order to test the results of compliance checking (section 4.4.1.). All results indicate correct modeling of the building code.

4.4.1. Test Cases

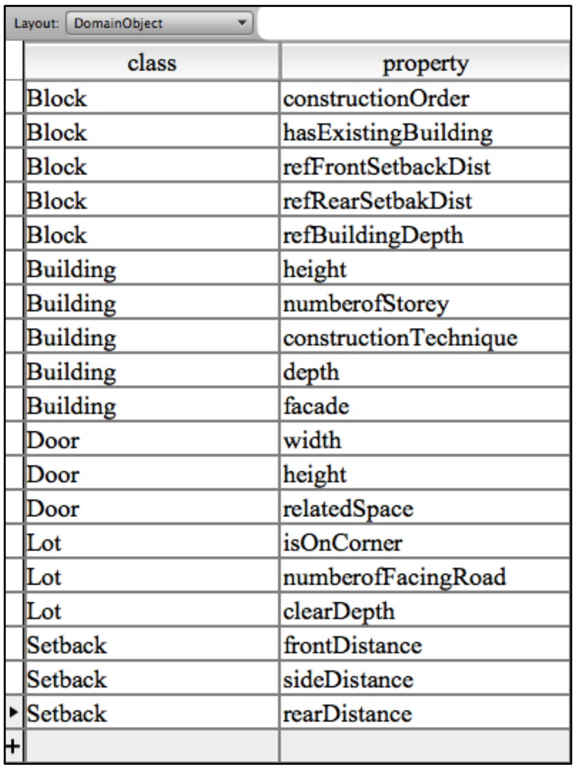
The usage and testing of the compliance checking prototype is illustrated in this section by a set of checking scenarios. Clause 27 (distance of setbacks), 28 (depth of buildings), 29 (façade of buildings), 30 (height of buildings), and 47 (doors) of IMHZCode are used as demonstrative examples to show how building projects can be checked for their compliance with building codes. These clauses are given in appendix A. However, other clauses and building models can be similarly handled by the prototype for different compliance checking scenarios.

A relational database application, FileMaker Pro, is used to create and store the sample code model. The code model database is directly and easily usable by the building code authors who have no programming background for modification of the content. Whenever needed, the user can access the database to revise or extend the code model by adding new rules and relationships. The creation of the sample code model includes the following steps based on the new proposed model:

- Create domain object and concept mapping tables (Figure 4.9 and Figure 4.10)
- Create rule table (Figure 4.11)

- Classify rule instances that are associated with the same property of the same concept and create rule-set table (Figure 4.12).
- Classify rule-set instances based on the concept they are related to and create rule-set group table (Figure 4.13).

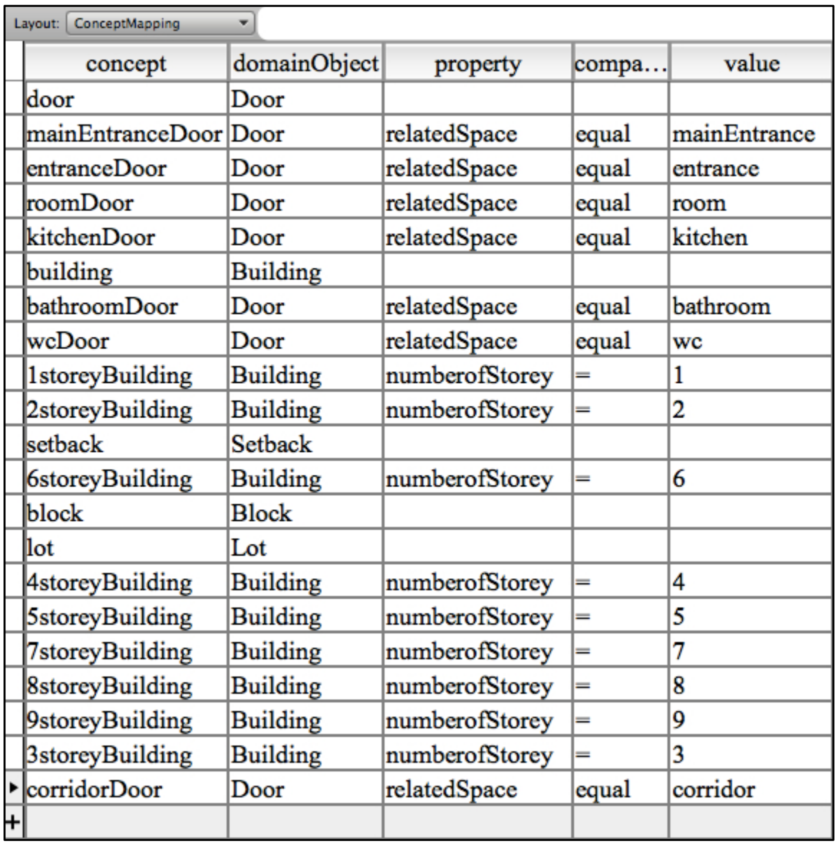
The DomainObject table shown in Figure 4.9 holds information on domain objects and their properties that represent identified entities (terms) in the sample building code text. Block, Lot, Setback, Building, and Door objects have the specified properties for the sample building code model domain. These domain object and property pairs (each record in the DomainObject table) are required when building rule objects. They are used by multiple records in the Rule table as the subject part (concept-property) of the *requirement* construct when creating rules.



class	property
Block	constructionOrder
Block	hasExistingBuilding
Block	refFrontSetbackDist
Block	refRearSetbakDist
Block	refBuildingDepth
Building	height
Building	numberofStorey
Building	constructionTechnique
Building	depth
Building	facade
Door	width
Door	height
Door	relatedSpace
Lot	isOnCorner
Lot	numberofFacingRoad
Lot	clearDepth
Setback	frontDistance
Setback	sideDistance
Setback	rearDistance

Figure 4.9. Domain object table of the sample code model

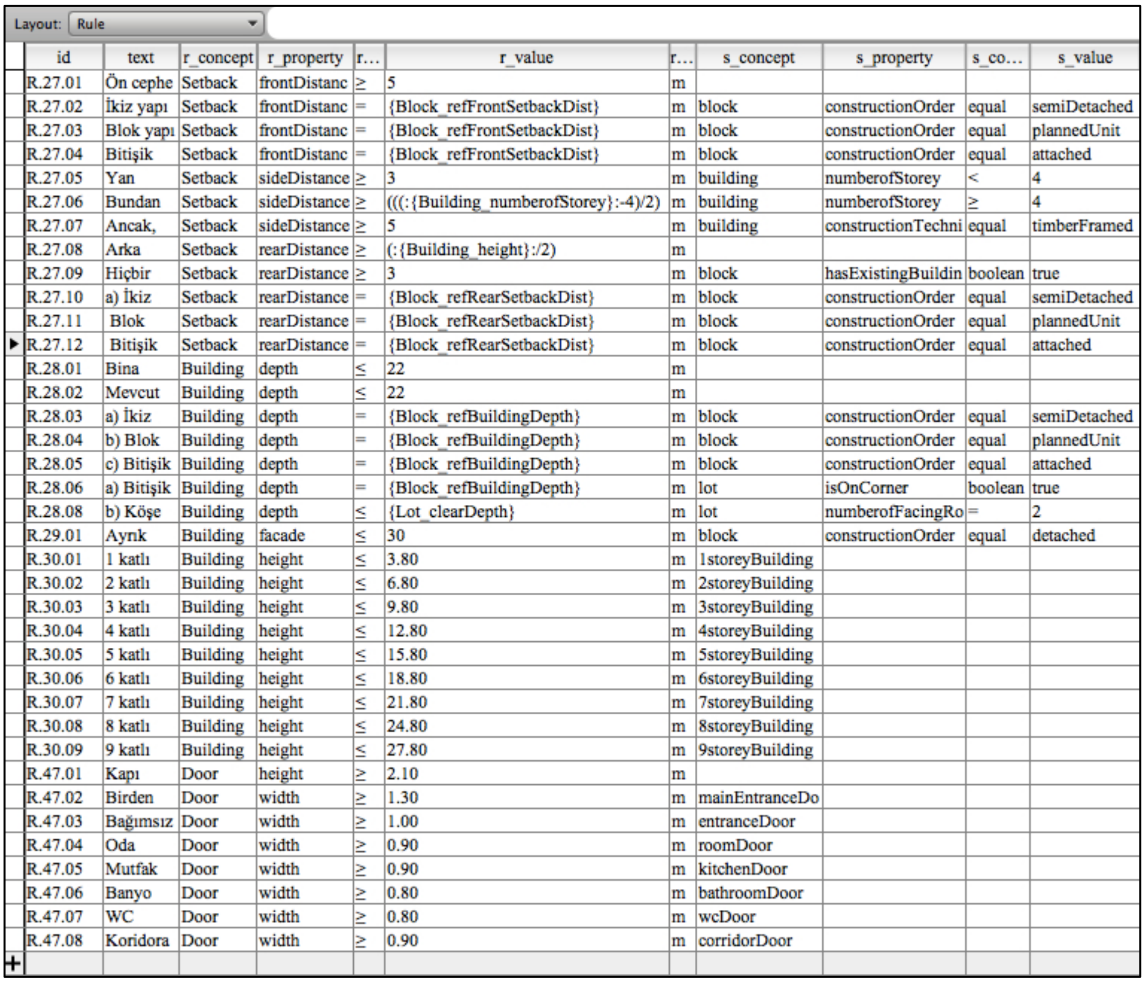
In addition to the DomainObject table, ConceptMapping table shown in Figure 4.10 also holds information required for defining rules. The ConceptMapping table lists all concepts referred to in the building code text. Each concept is either a domain object or a filtered (a specified property is checked against specified criteria) set of domain objects. When creating rules, the *selection* construct of each rule is a concept from the ConceptMapping table that can be further filtered. The DomainObject and ConceptMapping tables together form the lowest level of the developed representation model which is described in section 3.4.1.



concept	domainObject	property	compa…	value
door	Door			
mainEntranceDoor	Door	relatedSpace	equal	mainEntrance
entranceDoor	Door	relatedSpace	equal	entrance
roomDoor	Door	relatedSpace	equal	room
kitchenDoor	Door	relatedSpace	equal	kitchen
building	Building			
bathroomDoor	Door	relatedSpace	equal	bathroom
wcDoor	Door	relatedSpace	equal	wc
1storeyBuilding	Building	numberofStorey	=	1
2storeyBuilding	Building	numberofStorey	=	2
setback	Setback			
6storeyBuilding	Building	numberofStorey	=	6
block	Block			
lot	Lot			
4storeyBuilding	Building	numberofStorey	=	4
5storeyBuilding	Building	numberofStorey	=	5
7storeyBuilding	Building	numberofStorey	=	7
8storeyBuilding	Building	numberofStorey	=	8
9storeyBuilding	Building	numberofStorey	=	9
3storeyBuilding	Building	numberofStorey	=	3
corridorDoor	Door	relatedSpace	equal	corridor

Figure 4.10. Concept-mapping table of the sample code model

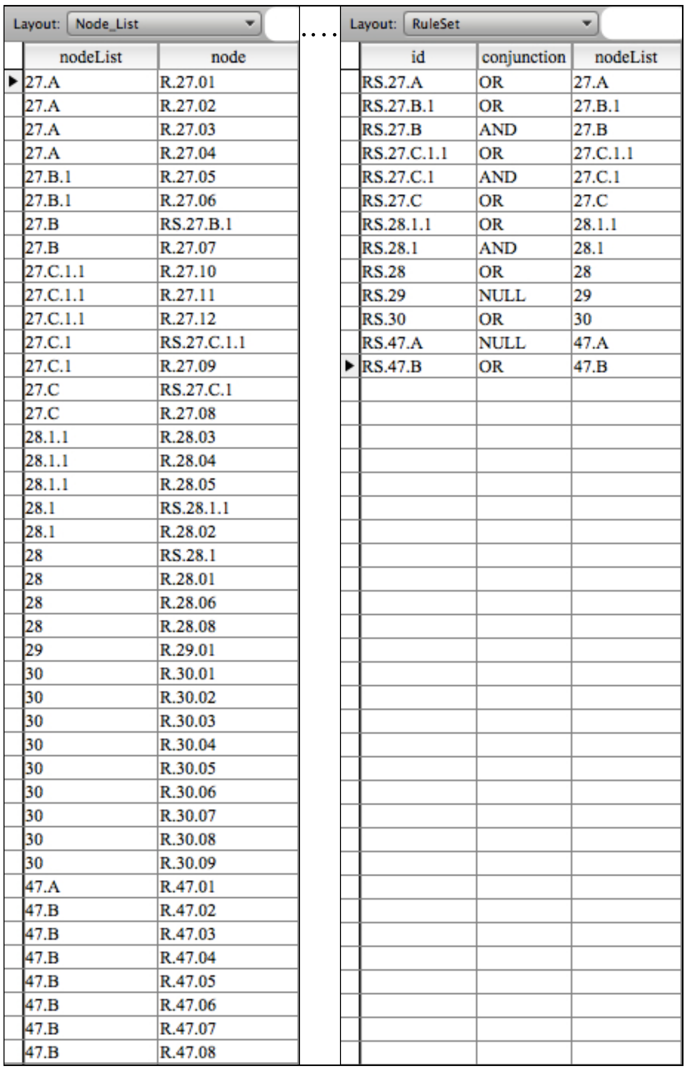
The Rule table shown in Figure 4.11 holds information on individual rule objects of the sample code model. 37 rule records are entered for the clauses selected for the test cases. For each rule record, the subject part (concept and property) of the *requirement* construct comes from the DomainObject table while the subject part of the *selection* construct comes from the ConceptMapping table.


Layout: Rule

id	text	r_concept	r_property	r…	r_value	r…	s_concept	s_property	s_co…	s_value
R.27.01	Ön cephe	Setback	frontDistanc	≥	5	m				
R.27.02	İkiz yapı	Setback	frontDistanc	=	{Block_refFrontSetbackDist}	m	block	constructionOrder	equal	semiDetached
R.27.03	Blok yapı	Setback	frontDistanc	=	{Block_refFrontSetbackDist}	m	block	constructionOrder	equal	plannedUnit
R.27.04	Bitişik	Setback	frontDistanc	=	{Block_refFrontSetbackDist}	m	block	constructionOrder	equal	attached
R.27.05	Yan	Setback	sideDistance	≥	3	m	building	numberofStorey	<	4
R.27.06	Bundan	Setback	sideDistance	≥	(((:{Building_numberofStorey}:-4)/2)	m	building	numberofStorey	≥	4
R.27.07	Ancak,	Setback	sideDistance	≥	5	m	building	constructionTechni	equal	timberFramed
R.27.08	Arka	Setback	rearDistance	≥	(:{Building_height}:/2)	m				
R.27.09	Hiçbir	Setback	rearDistance	≥	3	m	block	hasExistingBuildin	boolean	true
R.27.10	a) İkiz	Setback	rearDistance	=	{Block_refRearSetbackDist}	m	block	constructionOrder	equal	semiDetached
R.27.11	Blok	Setback	rearDistance	=	{Block_refRearSetbackDist}	m	block	constructionOrder	equal	plannedUnit
R.27.12	Bitişik	Setback	rearDistance	=	{Block_refRearSetbackDist}	m	block	constructionOrder	equal	attached
R.28.01	Bina	Building	depth	≤	22	m				
R.28.02	Mevcut	Building	depth	≤	22	m				
R.28.03	a) İkiz	Building	depth	=	{Block_refBuildingDepth}	m	block	constructionOrder	equal	semiDetached
R.28.04	b) Blok	Building	depth	=	{Block_refBuildingDepth}	m	block	constructionOrder	equal	plannedUnit
R.28.05	c) Bitişik	Building	depth	=	{Block_refBuildingDepth}	m	block	constructionOrder	equal	attached
R.28.06	a) Bitişik	Building	depth	=	{Block_refBuildingDepth}	m	lot	isOnCorner	boolean	true
R.28.08	b) Köşe	Building	depth	≤	{Lot_clearDepth}	m	lot	numberofFacingRo	=	2
R.29.01	Ayrık	Building	facade	≤	30	m	block	constructionOrder	equal	detached
R.30.01	1 katlı	Building	height	≤	3.80	m	1storeyBuilding			
R.30.02	2 katlı	Building	height	≤	6.80	m	2storeyBuilding			
R.30.03	3 katlı	Building	height	≤	9.80	m	3storeyBuilding			
R.30.04	4 katlı	Building	height	≤	12.80	m	4storeyBuilding			
R.30.05	5 katlı	Building	height	≤	15.80	m	5storeyBuilding			
R.30.06	6 katlı	Building	height	≤	18.80	m	6storeyBuilding			
R.30.07	7 katlı	Building	height	≤	21.80	m	7storeyBuilding			
R.30.08	8 katlı	Building	height	≤	24.80	m	8storeyBuilding			
R.30.09	9 katlı	Building	height	≤	27.80	m	9storeyBuilding			
R.47.01	Kapı	Door	height	≥	2.10	m				
R.47.02	Birden	Door	width	≥	1.30	m	mainEntranceDo			
R.47.03	Bağımsız	Door	width	≥	1.00	m	entranceDoor			
R.47.04	Oda	Door	width	≥	0.90	m	roomDoor			
R.47.05	Mutfak	Door	width	≥	0.90	m	kitchenDoor			
R.47.06	Banyo	Door	width	≥	0.80	m	bathroomDoor			
R.47.07	WC	Door	width	≥	0.80	m	wcDoor			
R.47.08	Koridora	Door	width	≥	0.90	m	corridorDoor			

Figure 4.11. Rule table of the sample code model

The Node_List and RuleSet tables shown in Figure 4.12 together hold the information on the rule-set trees. Each rule-set tree brings all rule records associated with the same property of the same concept. Lastly, the RuleSet_Group table (Figure 4.13) holds information on various organizations of rule-sets. For the test cases there is only a single organization. Rule-sets are grouped based on the domain object they are related to.


Layout: Node_List

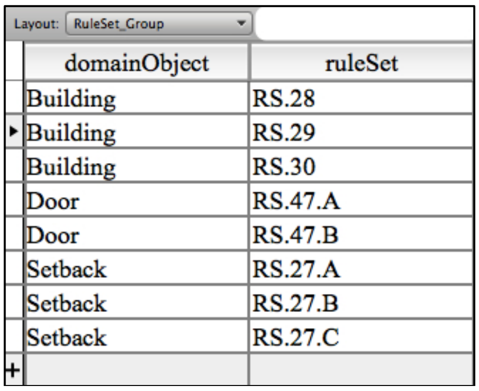
nodeList	node
27.A	R.27.01
27.A	R.27.02
27.A	R.27.03
27.A	R.27.04
27.B.1	R.27.05
27.B.1	R.27.06
27.B	RS.27.B.1
27.B	R.27.07
27.C.1.1	R.27.10
27.C.1.1	R.27.11
27.C.1.1	R.27.12
27.C.1	RS.27.C.1.1
27.C.1	R.27.09
27.C	RS.27.C.1
27.C	R.27.08
28.1.1	R.28.03
28.1.1	R.28.04
28.1.1	R.28.05
28.1	RS.28.1.1
28.1	R.28.02
28	RS.28.1
28	R.28.01
28	R.28.06
28	R.28.08
29	R.29.01
30	R.30.01
30	R.30.02
30	R.30.03
30	R.30.04
30	R.30.05
30	R.30.06
30	R.30.07
30	R.30.08
30	R.30.09
47.A	R.47.01
47.B	R.47.02
47.B	R.47.03
47.B	R.47.04
47.B	R.47.05
47.B	R.47.06
47.B	R.47.07
47.B	R.47.08

....

Layout: RuleSet

id	conjunction	nodeList
RS.27.A	OR	27.A
RS.27.B.1	OR	27.B.1
RS.27.B	AND	27.B
RS.27.C.1.1	OR	27.C.1.1
RS.27.C.1	AND	27.C.1
RS.27.C	OR	27.C
RS.28.1.1	OR	28.1.1
RS.28.1	AND	28.1
RS.28	OR	28
RS.29	NULL	29
RS.30	OR	30
RS.47.A	NULL	47.A
RS.47.B	OR	47.B

Figure 4.12. Classification of rules and rule-set table of sample code model


Layout: RuleSet_Group

domainObject	ruleSet
Building	RS.28
Building	RS.29
Building	RS.30
Door	RS.47.A
Door	RS.47.B
Setback	RS.27.A
Setback	RS.27.B
Setback	RS.27.C

Figure 4.13. Rule-set group table of the sample code model

The tree model of the sample code model is illustrated in Figure 4.14. As can be seen from this figure, the sample code model for Clause 27, 28, 29, 30, and 47 of IMHZCode consist of rule-sets related to Door, Setback, and Building objects. 37 rules are organized in 8 rule-sets and 3 rule-set groups. During code compliance checking, each setback, building, and door object in the given building project will be checked against applicable rules under related rule-sets. For the test cases, the prototype implementation makes 2 rule-set checks for each door, 3 rule-set checks for the building and 3 rule-set checks its setback.

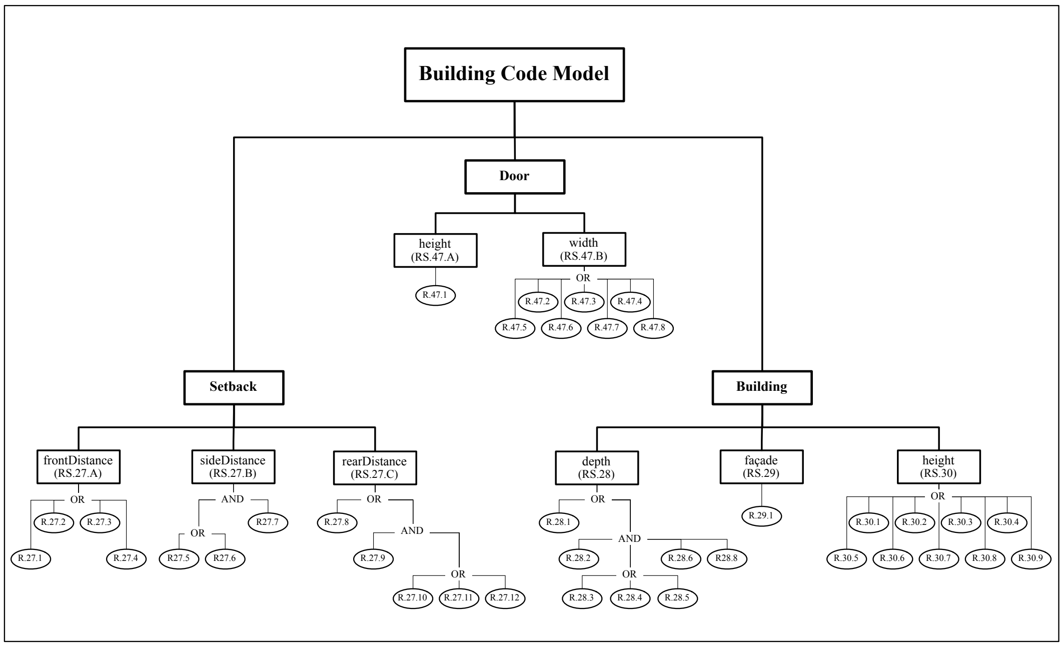


Figure 4.14. Tree model of the sample code model

As a test case for the compliance checking prototype, a residential building project is used. The plan of the 3 storey apartment building is shown in Figure 4.15. This sample building is modeled using a BIM system, Graphisoft Archicad, and exported as an IFC file. As explained in section 3.5.3 and section 4.3.2, the building information models created in a BIM environment do not include all needed information for code compliance checking. Therefore, for the test case, required information which cannot be obtained from the raw Archicad BIM model is added to the model as property-set (PSet) extensions of the IFC schema (Figure 4.16). The domain object table (Figure 4.9) of the sample code model gives required BIM information for the code.

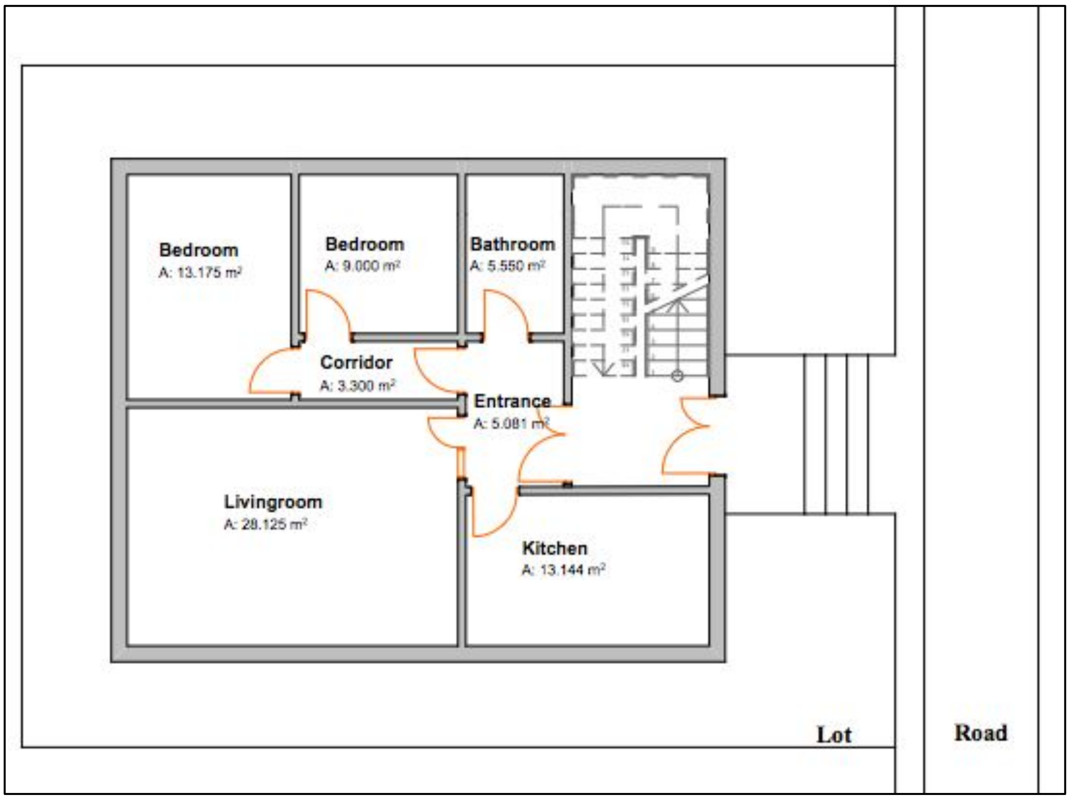
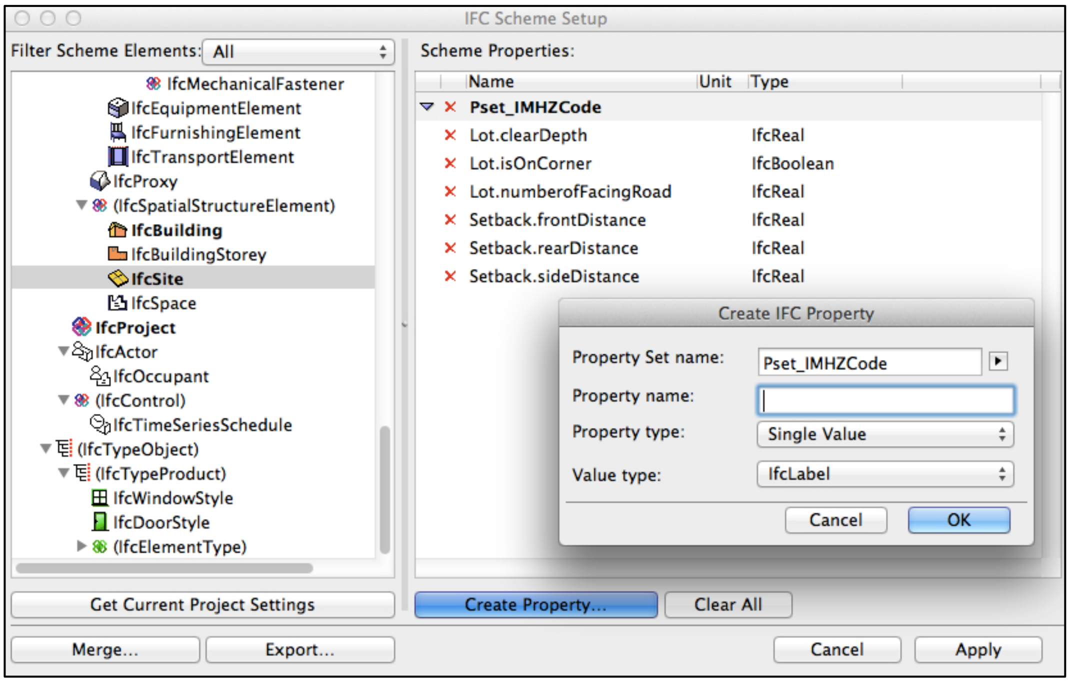


Figure 4.15. Sample building model



Name	Unit	Type
Pset_IMHZCode		
Lot.clearDepth		IfcReal
Lot.isOnCorner		IfcBoolean
Lot.numberofFacingRoad		IfcReal
Setback.frontDistance		IfcReal
Setback.rearDistance		IfcReal
Setback.sideDistance		IfcReal

Create IFC Property

Property Set name: Pset_IMHZCode
Property name:
Property type: Single Value
Value type: IfcLabel

Figure 4.16. Creation of missing information as a PSet extension of IFC schema

The compliance checking prototype has been tested through various checking scenarios. Three demonstrative test cases are presented here. For all scenarios the IFC file of the sample building project is imported to the prototype for compliance checking against the sample IMHZCode model. The three cases being presented conduct compliance checks of the 3-storey residential building design with rules related to the building, its setback, and doors. For demonstration doors have been selected since rules on doors are representative of the majority of the clauses in IMHZCode. Setbacks have been selected because Clause-27 that defines the rules on setback distances is the most complex clause in IMHZCode. Buildings have been selected because rules on buildings are not contained in a single clause but are distributed among three different clauses (Clause-28, Clause-29, Clause-30).

Case 1 is the base case. In this case, the *construction order* of the city block where the site is located is defined as *detached* and the *construction technique* of the building is set to *concrete*. Case 2 is the same design but the construction order of the block is changed to *semi-detached*. The setback rule that applies to case 2 is therefore different than the base case. Case 3 introduces a second change and the construction technique is defined as *timber framed*. The setback rule that applies to case 3 is different than both previous cases. The prototype is able to correctly handle these cases.

In case 1 (related design parameters are given in Table 4.14), the rearDistance property of the setback object (S001) is not valid according to Rule-set RS.27.C. Rule-set RS.27.C includes five rules that indicate minimum rear distance of setbacks. Which rule is applicable depends on the construction order of the block where the building is to be built. For case 1, the applicable rule for blocks with detached order, states that rear setbacks need to be at least half of the total building height (Rule R.27.08). Since the design does not meet this requirement, case 1 fails the check. During processing, the system first finds the applicable rule from the related rule-set then applies this rule to the object (setback object). For case 1, the door object D006 is also not valid. It does not pass the check on rule-set RS.47.B (related to door widths) because of rule R.47.06. There are seven rules restricting minimum door width. Because the D006 is a bathroom door, the system applies R.47.06, which is the correct rule for bathroom doors. The checking result for case 1 is given in Table 4.15

Table 4.14. Design parameters for Case 1

Project: Case 1

Block

id	*constrOrder*	*hasExistingBuilding*	*refFrontSetbackDis*	*refRearSetbackDis*	*refBuildingDepth*
BL01	detached	true	5.00	3.00	14.00

Lot

id	*isOnCorner*	*clearDepth*	*numberofFacingRoad*
L001	false	1	20.00

Setback

id	*frontDistance*	*sideDistance*	*rearDistance*
S001	5.00	3.00	3.00

Building

id	*depth*	*façade*	*height*	*constructionTech*	*numberofStorey*
B001	11.50	9.50	9.30	concrete	3

Door

id	*relatedSpace*	*height*	*width*
D001	mainEntrance	2.30	1.50
D002	entrance	2.30	1.30
D003	room	2.10	1.20
D004	kitchen	2.10	0.90
D005	corridor	2.10	0.90
D006	bathroom	2.10	0.70
D007	room	2.10	0.90
D008	room	2.10	0.90
D0xx	………	……	……

Table 4.15. Checking result for case 1

Checking Result: Case 1

	Setback			Building			Door	
	frontDistance (RS.27.A)	*sideDistance (RS.27.B)*	*rearDistance (RS.27.C)*	*depth (RS.28)*	*façade (RS.29)*	*height (RS.30)*	*height (RS.47.A)*	*width (RS.47.B)*
S001	√ (R.27.01)	√ (R.27.05)	**X** (R.27.08)	-	-	-	-	-
B001	-	-	-	√ (R.28.01)	√ (R.29.01)	√ (R.30.03)	-	-
D001	-	-	-	-	-	-	√ (R.47.01)	√ (R.47.02)
D002	-	-	-	-	-	-	√ (R.47.01)	√ (R.47.03)
D003	-	-	-	-	-	-	√ (R.47.01)	√ (R.47.04)
D004	-	-	-	-	-	-	√ (R.47.01)	√ (R.47.05)
D005	-	-	-	-	-	-	√ (R.47.01)	√ (R.47.08)
D006	-	-	-	-	-	-	√ (R.47.01)	**X** (R.47.06)
D007	-	-	-	-	-	-	√ (R.47.01)	√ (R.47.04)
D008	-	-	-	-	-	-	√ (R.47.01)	√ (R.47.04)
D0xx	….	….	….	….	….	….	…	…

For case 2 with semi-detached ordered block (Table 4.16) the same setback object S001 is valid according to Rule-set RS.27.C based on rules R.27.09 and R.27.10. In Clause-27 of IMHZCode, it is stated that "in Semi-Detached Building Blocks, if there is an existing building in one of the two lots then rear setbacks will be determined based on the existing building with the condition that rear setbacks will never be less than 3.00 meters". In the code model, this rule statement is represented as two separate rule objects (R.27.09 and R.27.10) that are connected to each other with an "AND" conjunction and put into a rule-set (RS.27.C). Again, the system applied the correct rules to the object and returned the correct result (Table 4.17).

Table 4.16. Design parameters for Case 2

Project: Case 2					
Block					
id	*constrOrder*	*hasExistingBuilding*	*refFrontSetbackDis*	*refRearSetbackDis*	*refBuildingDepth*
BL01	semiDetached	true	5.00	3.00	14.00
Lot					
id	*isOnCorner*	*clearDepth*	*numberofFacingRoad*		
L001	false	20.00	1		
Setback					
id	*frontDistance*	*sideDistance*	*rearDistance*		
S001	5.00	3.00	3.00		
Building					
id	*depth*	*façade*	*height*	*constructionTech*	*numberofStorey*
B001	11.50	9.50	9.30	concrete	3
….	….	…..	….	….	….

Table 4.17. Checking result for case 2

Checking Result: Case 2							
	Setback			**Building**			**….**
	frontDistance (RS.27.A)	*sideDistance (RS.27.B)*	*rearDistance (RS.27.C)*	*depth (RS.28)*	*façade (RS.29)*	*height (RS.30)*	…..
S001	√ (R.27.01)	√ (R.27.05)	√ (R.27.09 & R.27.10)	-	-	-	…..
B001	-	-	-	√ (R.28.01)	√ (R.29.01)	√ (R.30.03)	….
……	….	….	….	…..	….	….	….

For case 3 which is a timber frame construction on a semi-detached ordered block (Table 4.18) the same setback object is again not valid. While case 1 it fails the checks due to the rear setback distance, case 3 fails due to side setback distance. The same setback object that passed the checks against rule-set RS.27.B in the first two cases, fails after the construction technique of the building is specified as being timber framed. Part B of Clause-27 in IMHZCode states that "Side setbacks (up to and including 4 storeys) shall be 3.00 meters. For buildings taller than this side setbacks are increased by 0.5 meters for every additional storey. However, for timber-framed buildings side setbacks must be at least 5.00 meters." In the code model, these three rule statements are represented as three rule objects (R.27.5, R.27.6, and R.27.7). Rules

R.27.5 and R.27.6 are connected to each other with an "OR" conjunction and nested as a sub-set of rule-set RS.27.B. This subset is connected to Rule R.27.7 with an "AND" conjunction and form rule-set RS.27.B. The system checks setback S001 against the first rule (R.27.5) of the rule-set RS.27.B and obtains a pass result. The system skips the check for R.27.6 since the "OR" conjunction that joins these two rules does not necessitate further checking. After the system concludes that the setback object passes this "OR" sub-set, it continues the checking process with R.27.7 because RS.27.B is an "AND" rule-set requiring a pass result from all conditions it joins. The setback object fails its check against R.27.7 due to its shorter than rule-specified distance and thus fails the overall check. The checking result of the case 3 is given Table 4.19.

Table 4.18. Design parameters for Case 3

Project: Case 3

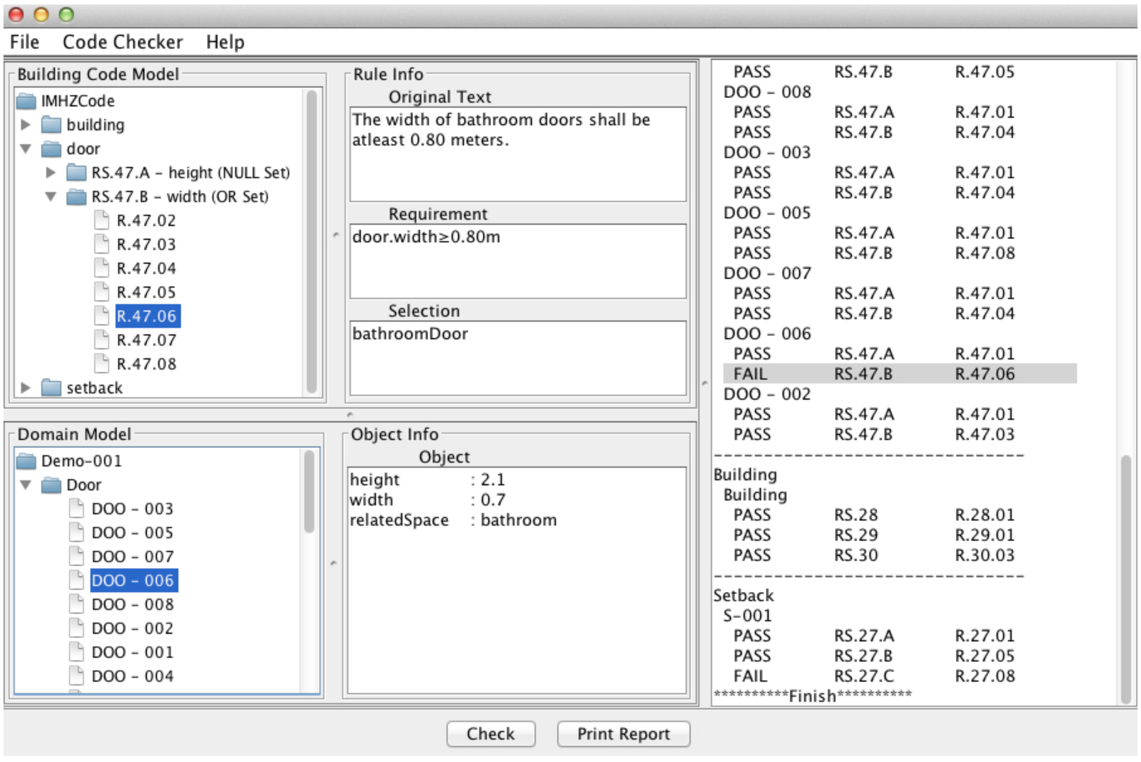
Block					
id	*constrOrder*	*hasExistingBuilding*	*refFrontSetbackDis*	*refRearSetbackDis*	*refBuildingDepth*
BL01	semiDetached	true	5.00	3.00	14.00
Lot					
id	*isOnCorner*	*clearDepth*	*numberofFacingRoad*		
L001	false	20.00	1		
Setback					
id	*frontDistance*	*sideDistance*	*rearDistance*		
S001	5.00	3.00	3.00		
Building					
id	*depth*	*façade*	*height*	*constructionTech*	*numberofStorey*
B001	11.50	9.50	9.30	timberFramed	3
….	….	…..	….	….	….

Table 4.19. Checking result for case 3

Checking Result: Case 3

	Setback			**Building**			**….**
	frontDistance (RS.27.A)	*sideDistance (RS.27.B)*	*rearDistance (RS.27.C)*	*depth (RS.28)*	*façade (RS.29)*	*height (RS.30)*	…..
S001	√ (R.27.01)	**X** (R.27.07)	√ (R.27.09 & R.27.10)	-	-	-	…..
B001	-	-	-	√ (R.28.01)	√ (R.29.01)	√ (R.30.03)	….
……	….	….	….	…..	….	….	….

Figure 4.17 is a screenshot where compliance checking of the sample building against IMHZCode clauses related to setback, door and building concepts takes place. Case 1 is active in the screenshot.



Building Code Model

```
IMHZCode
▶ building
▼ door
    ▶ RS.47.A – height (NULL Set)
    ▼ RS.47.B – width (OR Set)
        R.47.02
        R.47.03
        R.47.04
        R.47.05
        R.47.06
        R.47.07
        R.47.08
▶ setback
```

Rule Info

Original Text: The width of bathroom doors shall be atleast 0.80 meters.

Requirement: door.width≥0.80m

Selection: bathroomDoor

Domain Model

```
Demo-001
▼ Door
    DOO – 003
    DOO – 005
    DOO – 007
    DOO – 006
    DOO – 008
    DOO – 002
    DOO – 001
    DOO – 004
```

Object Info

Object

```
height          : 2.1
width           : 0.7
relatedSpace    : bathroom
```

```
  PASS        RS.47.B       R.47.05
 DOO – 008
  PASS        RS.47.A       R.47.01
  PASS        RS.47.B       R.47.04
 DOO – 003
  PASS        RS.47.A       R.47.01
  PASS        RS.47.B       R.47.04
 DOO – 005
  PASS        RS.47.A       R.47.01
  PASS        RS.47.B       R.47.08
 DOO – 007
  PASS        RS.47.A       R.47.01
  PASS        RS.47.B       R.47.04
 DOO – 006
  PASS        RS.47.A       R.47.01
  FAIL        RS.47.B       R.47.06
 DOO – 002
  PASS        RS.47.A       R.47.01
  PASS        RS.47.B       R.47.03
--------------------------------
Building
 Building
  PASS        RS.28         R.28.01
  PASS        RS.29         R.29.01
  PASS        RS.30         R.30.03
--------------------------------
Setback
 S-001
  PASS        RS.27.A       R.27.01
  PASS        RS.27.B       R.27.05
  FAIL        RS.27.C       R.27.08
**********Finish**********
```

Figure 4.17. A screenshot of the testing application

CHAPTER 5

CONCLUSION

In this dissertation a new representation model for building codes has been developed and an accompanying modeling methodology has been proposed. The new representation model and methodology describing a formal modeling process, allows representing domain concepts, individual rule statements, relationships between rules, and alternative methods of structuring (organizing) the overall building code document in a computable format to be used by future automated compliance checking systems. This chapter provides a brief summary and discusses contributions of this dissertation along with directions for future work.

5.1. Findings and Discussion

This dissertation proposed and demonstrated a new representation model establishing a methodology for representation of building codes as presented in detail in section 3.4. This new model divides the representation into four levels to provide a systematic structure for building codes in computational form. These levels are;

1. Domain level for representing concepts that appear in the building code.
2. Rule level for representing individual rule statements of the building code.
3. Rule-set level for representing inter-relations between the rule statements.
4. Management level for representing the organization of the building code.

For creating digital versions of building codes, based on the new representation model, a building code modeling methodology is defined. The methodology is comprised of three stages: analysis, representation, implementation.

For more than two decades, the research community has been investigating automated code compliance checking and proposed several methods of representation for modeling building codes. Several researchers have suggested various models and automated building code checking environments based on these models (Fenves, 1966; Fenves et al., 1987; Jain et al., 1989; Rasdorf & Lakmazaheri, 1990; Garrett & Hakim, 1992; Yabuki & Law, 1993; Kiliccote et al., 1994; Wix, 2008; Eastman, 2009; Yurchyshyna & Zarli, 2009; Pauwels et al., 2011). However, there has been limited success in transferring these environments into practice (Hakim & Garrett, 1992; Fenves et al., 1995). 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). Building codes have been modeled using various methods such as decision tables, hard-coded rules, programming logic, domain-specific rule language, semantic modeling. Each of these methods has its own inherent limitations. Ideally, the building code model 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 new representation model developed in this dissertation aims to meet the above requirements.

The main contribution of this dissertation is a new model establishing a methodology for representing building codes structured in four levels, as explained in CHAPTER 3 in detail. The new representation model is theoretically grounded in Nyman and Fenves (1975)'s work on an abstract model of the logical structure of building codes and combines it with an adapted version of SMARTcodes' semantic-oriented representation of building codes (Conover, 2009) for modeling rule statements.

Although Fenves and Nyman's work provides a solid theoretical foundation for representation of building codes, previous building code models based on this theory were not widely adopted in AEC industry mainly due to the immature information technologies for knowledge representation at the time. Representation methods in the modeling of building code information were inefficient and not easy understandable by non-programmer users, therefore code models quickly became hard to maintain and build. The four-level structure they introduced is still applicable and has proven to be a robust method for decomposing building codes.

SMARTcodes' semantic-oriented representation approach, which is a relatively new method for knowledge representation, provides an easy to understand, elegant method for modeling rule statements. It utilizes a simple scheme (RASE constructs) that is applicable for all types of rule statements, and with it, non-programmer users are able to build and maintain building code models. A pilot study carried out in the early phase of the research showed that it can be utilized for representing building codes, however, several shortcomings were identified. 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. This shortcoming was addressed by the first level of the new representation model creating domain objects of interest and concept mapping table that are used to define repeating concepts once. Second shortcoming is the lack of explicit relationships between individual rule statements. In SMARTcodes' approach, relationship representation (hierarchy within rules) is handled separately in 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 third level of the new model addresses this shortcoming by allowing to build the logical hierarchy relationships between rules. As a result, while SMARTcode's approach was adopted, it was modified into a four level representation that improves it by eliminating redundancies and adding logical relationships.

The dissertation 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 format independent of automated compliance checking systems. The four levels are: The domain level, the rule level, the rule-set level and the management level.

The domain level defines concepts in code checking (e.g. setback, eave, cantilever, etc.) as well as specialized object definitions (e.g. bathroom door, air shaft, developed lot, etc.) and holds a view of the building project appropriate for code checking when the checking system processes the BIM input. Objects at the domain level are used at the rule level where requirements are defined for them. With this separation all rules use the same definition for a concept and *consistency* in modeling is ensured.

After the building code is decomposed into independent rule statements, they are modeled as rule objects in the form of structured data based on modified RASE constructs. The use of domain level objects in building the constructs eliminates redundancies and ensures *conciseness*. The dependencies between rule statements are not modeled at this level. Since these rule objects at this level are simple and independent, *completeness* of the representation can be guaranteed by comparing them to the list of rule statements identified during the analysis of the building code.

The dependencies between rule objects are modeled at the rule-set level. When different requirements (rule statements) are stated for the same object depending on different conditions, the logical relationship hierarchy between rules is modeled at this level. This clear demarcation simplifies modeling of individual rule statements and at the same time allows analyzing complex conditional statements independent of the requirement data. Since at this level, only logical relations are modeled, it is easier to detect contradictions and redundancies within the conditions set forth in the building code. Ensuring uniqueness and correctness of the model can be managed through the rule-set definitions at this level.

While a simple aggregation of rule-sets is enough to represent all of the requirements contained in the building-code, it is also necessary to model the various methods of organizing the code. One method is the original section – heading – sub-heading organization that is used by the code document itself. The management level provides the tools to model such alternative organization methods to persist along with the rest of the code. Automated compliance checking systems will need to handle multiple building codes published by different authorities. When rules from multiple building codes apply to the same objects, the scope and priority of rule-sets will need to be modeled. In order to support these operations across multiple building codes, the rule-set classification objects are provided at the management level.

When compared with the RASE methodology, *maintainability* is preserved and even improved. Since concepts, individual rule statements, relations between the rule statements and organization of the building code are separately represented, required changes to the code model due to future revisions of the building code can be localized and handled without affecting the automated checking system.

The second contribution of this dissertation is a building code modeling methodology defining the process steps in developing building code representations by utilizing the proposed representation model, as explained in section 3.5 in detail. The proposed building code modeling methodology is comprised of three process stages:

1. Analysis of the building code for defining what should be represented explicitly for the purposes of automated compliance checking and documenting how much of the building code can be modeled reliably.
2. Representation of the building code by utilizing the developed representation model.

3. Implementation of the building code model within a compliance checking application.

The third contribution of this dissertation comes with a thorough analysis of the building codes as explained in detail in section 2.1. It is essential to understand the characteristics of building code documents and the various types of information contained in them in order to develop an appropriate building code model. The analysis led to a framework for classification of different rule types that are commonly found in building code documents. Classification according to formalizability is necessary for figuring out how much of the code can benefit from automated compliance checking. More importantly classification according to self-containedness is needed for understanding higher-order relationships between rule statements. This classification study is an essential first step for creating a digital representation of any building code document. Previous models and approaches for representation of building codes, several compliance checking systems and current technologies for implementation of building code models are also examined in section 2.2 and 2.3 to assess their advantages and limitations.

The proposed representation model is validated for use in the building design domain through a conducted case study and a prototype implementation. The validation study focused on modeling IMHZCode (Izmir Municipality Housing and Zoning Code) and illustrating the use of the new representation within future compliance checking applications. The prototype implementation was completed and code compliance checking of demonstrative examples, testing the capabilities of the model was successfully carried out.

In summary, through the research reported on in this dissertation:

- A new representation model, based on a four level structure stemming from Fenves and Nyman's work and utilizing an adapted version of SMARTCodes' semantic oriented representation of rule statements, has been developed establishing a methodology for representation of building codes.
- A building code modeling methodology is defined for utilizing the new representation model.
- The Izmir Municipality Housing and Zoning Code has been analyzed in detail and a framework for classification of rules has been developed.

- The Izmir Municipality Housing and Zoning Code has been modeled following the developed methodology as a demonstrative case.
- A prototype representing the functionality of future automated compliance checking systems has been implemented. Demonstrative examples of building projects have successfully been checked for compliance with IMHZCode.

To conclude, the main contribution of this dissertation is the new representation model for building codes that would be utilized in the development of automated compliance checking systems. The new 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. This dissertation has demonstrated the importance of separating and making explicit, the representations for domain concepts, individual rule statements, relationships between rule statements and organization of the building code.

5.2. Future Work

The prototype that has been implemented has been successful in checking compliance with IMHZCode for the building projects that have been chosen as test cases. However, there are limitations to this proof of concept. First, only IMHZCode has been modeled and the prototype has been tested with only this model. Yet, it should be noted that the building code that was chosen was Izmir Municipality Housing and Zoning Code. IMHZCode, as a code that is actually in effect and belonging to a metropolis, 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 new model is designed with the intent to provide a solution for representing only formalizable rules. Semi-formalizable and non-formalizable rules 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.

The research presented in this dissertation is a step forward towards providing automation support for compliance checking of building projects. Many directions for future research can be identified.

- A user interface for code authors is required for adding or modifying rules/concepts.
- Building a domain model is manual work. It should be investigated if domain models can be automatically derived from original texts of building codes by applying advanced artificial intelligence and natural language processing techniques.
- How building information models should be extended to hold information requirements of various code domains in order to support automated reasoning about code compliance is another issue that will need to be resolved.

REFERENCES

AEC3. (2012). International Code Council Retrieved February, 23, 2013, from http://www.aec3.com/en/5/5_013_ICC.htm

buildingSMART. (2008a). Industry Foundation Classes (Ifc) Data Model Retrieved 14 May, 2014, from http://www.buildingsmart.org/standards/ifc

buildingSMART. (2008b). Model View Definitions Retrieved 3 June, 2014, from http://www.buildingsmart.org/standards/mvd

Cheng, C. P., Lau, G. T., Law, K. H., Pan, J., & Jones, A. (2009). Improving Access to and Understanding of Regulations through Taxonomies. *Government Information Quarterly, 26*(2), 238-245. doi: 10.1016/j.giq.2008.12.008

Conover, D. (2007). *Development and Implementation of Automated Code Compliance Checking in the U.S.* International Code Council.

Conover, D. (2009). Method and Apparatus for Automatically Determining Compliance with Building Regulations, Washington, DC, US Patent No. US 2009/0125283 A1.

Ding, L., Drogemuller, R., Rosenman, M., Marchant, D., & Gero, J. (2006). *Automating Code Checking for Building Designs – Designcheck.* Paper presented at the Clients Driving Innovation: Moving Ideas into Practice, Gold Coast, Queensland, Australia.

Dym, C. L., Henchey, R. P., Delis, E. A., & Gonick, S. (1988). A Knowledge-Based System for Automated Architectural Code Checking. *Computer-Aided Design, 20*(3), 137-145. doi: http://dx.doi.org/10.1016/0010-4485(88)90021-8

Eastman, C. (2006). Ifc Overview. *Chuck Eastman.*

Eastman, C. (2009). Automated Assessment of Early Concept Designs. *Architectural Design, 79*(2), 52-57. doi: 10.1002/ad.851

Eastman, C. M., Lee, J.-m., Jeong, Y.-s., & Lee, J.-k. (2008). Implementation of Automatic Circulation Checking Module: Georgia Tech. .

Eastman, C. M., Lee, J.-m., Jeong, Y.-s., & Lee, J.-k. (2009). Automatic Rule-Based Checking of Building Designs. *Automation in Construction, 18*(8), 1011-1033. doi: 10.1016/j.autcon.2009.07.002

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

Fenves, S. J. (1976). The Structure of Building Specifications. In ERIC (Ed.), *NBS Building Science Series 90* (pp. 92). Washington, DC.: National Bureau of Standards.

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*(1), 13-34.

Fenves, S. J., Gaylord, E. H., & Goel, S. K. (1969). Decision Table Formulation of the 1969 Aisc Specification *Civil Engineering Studies SRS-347*.

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.

Garrett, J. H., & Fenves, S. J. (1987). A Knowledge-Based Standards Processor for Structural Component Design. *Engineering with Computers, 2*(4), 219-238. doi: 10.1007/BF01276414

Garrett, J. H. J., & Hakim, M. M. (1992). Object-Oriented Model of Engineering Design Standards. *Journal of Computing in Civil Engineering, 6*(3), 323-347.

Gero, J. S. (1984). *Amubc System: Final Report*: University of Sydney.

Hakim, M. M., & Garrett, J. H. (1992). *Issues in Modelling and Processing Design Standards.* Paper presented at the The joint CIB Workshops on Computers and Information in Construction.

Hakim, M. M., & Garrett, J. H. (1993). A Description Logic Approach for Representing Engineering Design Standards. *Engineering with Computers, 9*(2), 108-124. doi: 10.1007/bf01199049

Han, C., Kunz, J., & Law, K. H. (1997). Making Automated Building Code Checking a Reality. *Facility Management Journal*, 22-28.

Han, C., Kunz, J. C., & Law, K. H. (2002). Compliance Analysis for Disabled Access. In J. a. A. K. E. William J. McIver (Ed.), *Advances in Digital Government Technology, Human Factors, and Policy* (pp. 149-163). Kluwer, Boston, MA.

Han, C. S., Kunz, J. C., & Law, K. H. (1998). Client/Server Framework for on-Line Building Code Checking. *Journal of Computing in Civil Engineering, 12*(4), 181.

Harris, J. R., & Wright, R. N. (1980). Organization of Building Standards: Systematic Techniques for Scope and Arrangement *NBS, Building Science Series 136* (pp. 267). Washington, D.C.: National Bureau of Standards.

Hjelseth, E. (2009). *Foundation for Development of Computable Rules*. Paper presented at the CIB W078 26TH INTERNATIONAL CONFERENCE.

Hjelseth, E. (2012). Converting Performance Based Regulations into Computable Rules in Bim Based Model Checking Software *Ework and Ebusiness in Architecture, Engineering and Construction* (pp. 461-469): CRC Press.

ISO. (2002). Iso 10303-21:2002 Industrial Automation Systems and Integration -- Product Data Representation and Exchange -- Part 21: Implementation Methods: Clear Text Encoding of the Exchange Structure.

ISO. (2004). Iso 10303-11:2004 Industrial Automation Systems and Integration -- Product Data Representation and Exchange -- Part 11: Description Methods: The Express Language Reference Manual.

ISO. (2013). Iso 16739:2013 Industry Foundation Classes (Ifc) for Data Sharing in the Construction and Facility Management Industries.

Jain, D., Law, K. H., & Krawinkler, H. (1989). *On Processing Standards with Predicate Calculus*. Paper presented at the Sixth Conference on Computing in Civil Engineering, Atlanta, Georgia.

Jotne. (1994). Express Data Manager Retrieved 14 May, 2014, from http://www.epmtech.jotne.com

Kerrigan, S., & Law, K. H. (2003). *Logic-Based Regulation Compliance-Assistance.* Paper presented at the 9th International Conference on Artificial Intelligence and Law Scotland, United Kingdom.

Khemlani, L. (2004). The Ifc Building Model: A Look under the Hood. *AECbytes Feature*, 1-10.

Khemlani, L. (2005). Corenet E-Plancheck: Singapore's Automated Code Checking System. *AECbytes*.

Kiliccote, H., & Garrett, J. H. (1998). Standards Modeling Language. *Journal of Computing in Civil Engineering, 12*(3), 129-135.

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.* Paper presented at the First ASCE Congress on Computing in Civil Engineering, Washington, D.C, June 1994.

Kumar, B. (1995). *Knowledge Processing for Structural Design*. Southampton, UK ;: Computational Mechanics Publications.

Lee, J. K., Eastman, C. M., Lee, J., Kannala, M., & Jeong, Y. S. (2010). Computing Walking Distances within Buildings Using the Universal Circulation Network. *ENVIRONMENT AND PLANNING B-PLANNING & DESIGN, 37*(4), 628-645. doi: 10.1068/b35124

Liebich, T. (2002). Ifc Overview: Status of Standardization. *Implementation, and Use of Model-Based Work Together with IFC, 5*, 1-16.

Liebich, T., Wix, J., Forester, J., & Qi, Z. (2002). *Speeding-up the Building Plan Approval - the Singapore E-Plan Checking Project Offers Automatic Plan Checking Based on Ifc.* Paper presented at the European Conferences on Product and Process Modelling (ECPPM) 2002 - eWork and eBusiness in Architecture, Engineering and Construction, Portoroz, Slovenia.

Macit, S., İlal, M. E., Günaydın, H. M., & Suter, G. (2013). *İzmi Municipality Housing and Zoning Code Analysis and Representation for Compliance Checking.* Paper presented at the 20th Workshop of the European Group for Intelligent Computing in Engineering, Vienna, Austria, 1-3 July.

Nisbet, N., Wix, J., & Conover, D. (2009). The Future of Virtual Construction and Regulation Checking *Virtual Futures for Design, Construction & Procurement* (pp. 241-250): Blackwell Publishing Ltd.

novaCITYNETS. (2000). About Fornax™ - Plancheck Expert Retrieved 14 May, 2014, from http://www.novacitynets.com/fornax/about.htm

Nyman, D. J., & Fenves, S. J. (1975). An Organization Model for Design Specifications. *Journal of structural Division ASCE, 101*(4), 697-716.

Nyman, D. J., Fenves, S. J., & Wright, R. N. (1973). Restructuring Study of the Aisc Specification *Civil Engineering Studies SRS-393* (Vol. 1). Urbana-Champaign: Department of Civil Engineering, University of Illinois Engineering Experiment Station.

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*(5), 506-518. doi: 10.1016/j.autcon.2010.11.017

Rasdorf, W., & Fenves, S. (1980). *Design Specification Representation and Analysis.* Paper presented at the 2nd ASCE Conference on Computing in Civil Engineering, Baltimore, Maryland, United States, June 9-13.

Rasdorf, W., & Wang, T. (1988). Generic Design Standards Processing in an Expert System Environment. *Journal of Computing in Civil Engineering, 2*(1), 68-87. doi: 10.1061/(ASCE)0887-3801(1988)2:1(68)

Rasdorf, W. J., & Lakmazaheri, S. (1990). Logic-Based Approach for Modeling Organization of Design Standards. *Journal of Computing in Civil Engineering, 4*(2), 102-123. doi: 10.1061/(ASCE)0887-3801(1990)4:2(102)

Rosenman, M. A., & Gero, J. S. (1985). Design Codes as Expert Systems. *Computer-Aided Design, 17*(9), 399-409. doi: http://dx.doi.org/10.1016/0010-4485(85)90287-8

Salama, D. M., & El-Gohary, N. M. (2011). Semantic Modeling for Automated Compliance Checking *Computing in Civil Engineering (2011)* (pp. 641-648).

Sing, T. F., & Zhong, Q. (2001). Construction and Real Estate Network (Corenet). *Facilities, 19*(11/12), 419-428.

Solibri. (1999). Solibri Model Checker Retrieved 14 May, 2014, from http://www.solibri.com/products/solibri-model-checker/

Vassileva, S. (2000). *An Approach of Constructing Integrated Client/Server Framework for Operative Checking of Building Code.* Paper presented at the Construction Information Technology 2000: Taking the construction industry into the 21st century, Reykjavik, Iceland, June 28-30.

Waard, M. d. (1992). *Computer Aided Conformance Checking.* Paper presented at the Computers and Building Standards Workshop, Montreal, Canada, May 1992.

Wix, J. (2008). *Bim Automated Code Checking Based on Smartcodes.* Paper presented at the BuildingSmart Forum.

Wix, J., & Conover, D. (2007). Capturing and Using Knowledge with Building Informatin Modeling (Keynote) *Information and Knowledge Management - Helping the Practitioner in Planning and Building. Proceedings of the Cib W102 3rd International Conference 2007* (pp. p.35-48). Stuttgart (Germany): Fraunhofer IRB Verlag.

Yabuki, N., & Law, K. H. (1993). An Object-Logic Model for the Representation and Processing of Design Standards. *Engineering with Computers, 9*(3), 133-159. doi: 10.1007/bf01206345

Yang, Q. Z., & Li, X. (2001). *Representation and Execution of Building Codes for Automated Code Checking.* Paper presented at the 9th International Conference on Computer Aided Architectural Design Futures 2001, Eindhoven, Netherlands, Jul 08-11.

Yurchyshyna, A., Faron-Zucker, C., Thanh, N. L., & Zarli, A. (2008). *Towards an Ontology-Enabled Approach for Modeling the Process of Conformity Checking in Construction.* Paper presented at the CAiSE Forum, Montpellier, France, June 18-20.

Yurchyshyna, A., & Zarli, A. (2009). An Ontology-Based Approach for Formalisation and Semantic Organisation of Conformance Requirements in Construction. [doi: DOI: 10.1016/j.autcon.2009.07.008]. *Automation in Construction, 18*(8), 1084-1098.

Zhang, J., & El-Gohary, N. M. (2011). *Automated Information Extraction from Construction-Related Regulatory Documents for Automated Compliance Checking.* Paper presented at the CIM W78-W102 Conference, Sophia Antipolis, France.

APPENDIX A

DECOMPOSITION AND CLASSIFICATION OF IMHZCODE

This appendix shows the decomposition of Izmir Municipality Housing and Zoning Code (IMHZCode) clauses that include rule statements pertinent to buildings and the classification of these rule statements.

	Textual Expressions of Clause 27	**Statement Type**
	Madde 27–Bahçe Mesafeleri *Id&heading*	
1	Yürürlükteki imar planı kararlarında bahçe mesafelerine ilişkin ölçüler belirtilmediği takdirde, bahçe mesafelerinin aşağıdaki koşullara göre belirlenmesi zorunludur.	Applicability con.
	A- Ön Bahçe Mesafeleri: *subheading*	
2	Ön cephe ve yol kenarlarına, yeşil sahaya ve otoparka rastlayan bahçe mesafeleri en az 5.00 m.dir.	Rule
3	Ancak, yapılaşmanın başladığı adalarda (ayrık yapı nizamı verilen adalar hariç) aşağıdaki koşullara göre, aynı ada yüzündeki mevcut yapılar dikkate alınarak bahçe mesafeleri tayin edilecektir.	Applicability con.
4	a) İkiz yapı nizamı verilen adalarda; parsellerden birinde mevcut bina var ise, ikizine verilecek bahçe mesafesinin tayininde mevcut yapı esas alınır.	Rule
5	b) Blok yapı nizamı verilen adalarda; aynı blok içindeki parsellerden herhangi birinde mevcut bina var ise, sadece bu bloktaki parsellerin bahçe mesafesinin tayininde mevcut binanın bahçesi esas alınır.	Rule
6	c) Bitişik yapı nizamı verilen adalarda; ada yüzünün %50'den fazlasının (sayı ve/veya taban alanı itibarıyla) yürürlükteki imar planına göre gabarisinde ruhsatlı olarak teşekkül etmiş olması halinde, bahçe mesafesinin tayininde aynı ada yüzündeki ve gabarideki en yakın mevcut binalar esas alınır.	Rule
	B-Yan Bahçe Mesafeleri *subheading*	
7	Yan bahçe mesafeleri (4 kata kadar 4 kat dahil) 3.00 m. olacaktır.	Rule
8	Bundan yüksek katlı binalarda yan komşu mesafeleri 3 m.ye her bir kat için 0.50 m. ilave edilmek suretiyle tespit olunur.	Rule
9	Ancak, ahşap karkas yapılar için en az 5 m. mesafe bırakılması şarttır.	Rule
	B-Arka Bahçe Mesafeleri *subheading*	
10	Arka bahçe mesafeleri H/2'dir.	Rule
11	Bina yüksekliği olan H'nin tespiti bu yönetmeliğin 30. maddesine göre yapılır.	Clarification
12	Arka bahçe mesafeleri bir yola cephesi olan, iki yola cephesi olan (köşebaşı parselleri) ve 3 yola cephesi olan köşebaşı parsellerinde de uygulanır.	Clarification
13	Hiçbir yerde 3.00 m.den az olmamak koşulu ile, yapılaşmanın başladığı adalarda (ayrık yapı nizamı verilen adalar hariç) aşağıdaki koşullara göre, aynı ada yüzündeki mevcut yapılar dikkate alınarak arka bahçe mesafesi tayin edilecektir.	Rule & Applicability con.
14	a) İkiz yapı nizamı verilen adalarda; parsellerden birinde mevcut bina var ise, ikizine verilecek arka bahçe mesafesinin tayininde mevcut yapı esas alınır.	Rule
15	b) Blok yapı nizamı verilen adalarda; aynı blok içindeki parsellerden herhangi birinde mevcut bina var ise, sadece bu bloktaki parsellerin arka bahçe mesafesinin tayininde mevcut binanın arka bahçesi esas alınır.	Rule
16	c) Bitişik yapı nizamı verilen adalarda; ada yüzünün %50'den fazlasının (sayı ve/veya taban alanı itibarıyla) yürürlükteki imar planına göre gabarisinde ruhsatlı olarak teşekkül etmiş olması halinde, arka bahçelerin tayininde aynı ada yüzündeki ve gabarideki en yakın mevcut binalar esas alınır.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C27	ST2	R27.1	Self-contained	Formalizable
	ST4	R27.2	Linked-explanatory	Formalizable
	ST5	R27.3	Linked-explanatory	Formalizable
	ST6	R27.4	Linked-explanatory	Formalizable
	ST7	R27.5	Self-contained	Formalizable
	ST8	R27.6	Linked-explanatory	Formalizable
	ST9	R27.7	Linked-explanatory	Formalizable
	ST10	R27.8	Self-contained	Formalizable
	ST13	R27.9	Linked-explanatory	Formalizable
	ST14	R27.10	Linked-explanatory	Formalizable
	ST15	R27.11	Linked-explanatory	Formalizable
	ST16	R27.12	Linked-explanatory	Formalizable

Textual Expressions of Clause 28		Statement Type
Madde 28– Bina Derinlikleri		*Id&heading*
1	Bina derinlikleri; hiçbir yerde 22.00 m.yi geçmemek ve arka bahçe mesafesi H/2 nin altına düşmemek kaydıyla, imar planı koşulları da dikkate alınarak hesaplanır.	Rule
2	Mevcut yapılaşmanın başladığı adalarda hiçbir yerde arka komşu sınırına 3.00 m.den fazla yaklaşmamak ve max. 22.00 m.yi geçmemek şartı ile aşağıdaki koşullara göre aynı ada yüzündeki mevcut yapılaşma dikkate alınarak hesaplanır.	Rule
3	a) İkiz yapı nizamı verilen adalarda; parsellerden birinde mevcut bina var ise, ikizine verilecek bina derinliğinin tayininde mevcut yapı esas alınır.	Rule
4	b) Blok yapı nizamı verilen adalarda; aynı blok içindeki parsellerden herhangi birinde mevcut bina var ise, sadece bu bloktaki parsellerin bina derinliklerinin tayininde mevcut binanın bina derinliği esas alınır.	Rule
5	c) Bitişik yapı nizamı verilen adalarda; adanın %50'den fazlasının (sayı ve/veya taban alanı itibarıyla) yürürlükteki imar planına göre gabarisinde ruhsatlı olarak teşekkül etmiş olması halinde, bina derinliklerinin tayininde aynı gabarideki mevcut binalar esas alınır.	Rule
6	a) Bitişik yapı nizamı verilen adalarda; köşe başına rastlayan parsellerde yapı derinliği parselin yüz aldığı yollar üzerindeki komşu parsellere verilecek derinliklere göre belirlenir.	Rule
7	Yapı adasında bu yollara verilen derinliklerin ada köşesindeki parsellerin ada içi boşluğu ile irtibatını kesecek ölçüde olmaması halinde, bina derinliği 30 m. yi aşmamak kaydıyla bu parsellere *nizami aydınlıkla* parsel tamamına yapı izni verilebilir.	Rule
8	b) Köşe başından başka iki yola cephesi bulunan ve varsa ön, arka bahçe mesafeleri çıktıktan sonraki ortalama derinliği 30.00 m.den az olan parsellerde bu derinliğe kadar yapı yapılabilir.	Rule
9	Toplam kitle derinliği 30. m. yi aşmamak komşu binalarla uyum sağlamak, arka bahçeleri bütünleştirecek şekilde parselde iki kitle düzenlemeye *belediyesi yetkilidir.*	Rule
10	c) Her türlü bölgede derinliği az olan parsellerde 3.00 m. arka bahçe mesafesi bırakıldığında bina derinliğinin 10.00 m. altına düşmesi halinde arka bahçe mesafesini 1.00 m. ye kadar azaltmaya *belediyesi yetkilidir*.	Rule
11	d) Ayrık nizamda ve yapı emsali verilmemiş adalarda; bahçe mesafesi içinde kalmak koşulu ile yapı boyutları 22.00 x 30.00 m.yi aşamaz.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C28	ST1	R28.1	Self-contained	Formalizable
	ST2	R28.2	Linked-explanatory	Formalizable
	ST3	R28.3	Linked-explanatory	Formalizable
	ST4	R28.4	Linked-explanatory	Formalizable
	ST5	R28.5	Linked-explanatory	Formalizable
	ST6	R28.6	Linked-explanatory	Formalizable
	ST7	R28.7	Linked-explanatory	Semi-formalizable
	ST8	R28.8	Linked-explanatory	Formalizable
	ST9	R28.9	Linked-explanatory	Non-formalizable
	ST10	R28.10	Linked-explanatory	Non-formalizable
	ST11	R28.11	Self-contained	Formalizable

Textual Expressions of Clause 29		Statement Type
Madde 29– Bina Cepheleri		*Id&heading*
1	Ayrık yapı nizamına tabi olan yerlerde yapılacak yapıların max. bina cephesi (30.00) m.dir.	Rule
2	Ayrık ve ikiz yapı nizamına tabi olan yerlerde, daha uygun çözüm yolları bulmak maksadı ile birkaç dar parseli birlikte mütalaa ederek o yer için tespit edilen yapı karakterine uyacak bir tertipten uzaklaşmamak kaydıyla, bina cepheleri toplamı (30.00) m.yi geçmeyen ikili veya üçlü bloklar teşkil etmeye *belediyesi yetkilidir*.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C29	ST1	R29.1	Self-contained	Formalizable
	ST2	R29.2	Self-contained	Non-formalizable

Textual Expressions of Clause 30		Statement Type
Madde 30– Bina Yükseklikleri		*Id&heading*
1	İmar planlarında gösterilen bina yüksekliklerinin veya kat adetlerinin birbirlerine tahvilleri aşağıdaki şekilde tespit edilir.	Clarification
2	1 katlı binaların yüksekliği maks. 3.80 m.	Rule
3	2 katlı binaların yüksekliği maks. 6.80 m.	Rule
4	3 katlı binaların yüksekliği maks. 9.80 m.	Rule
5	4 katlı binaların yüksekliği maks. 12.80 m.	Rule
6	5 katlı binaların yüksekliği maks. 15.80 m.	Rule
7	6 katlı binaların yüksekliği maks. 18.80 m.	Rule
8	7 katlı binaların yüksekliği maks. 21.80 m.	Rule
9	8 katlı binaların yüksekliği maks. 24.80 m.	Rule
10	9 katlı binaların yüksekliği maks.27.80 m.	Rule
11	10 katlı binaların yüksekliği maks.30.80 m.	Rule
12	Yeni yapılacak binalarda; tayin edilmiş ise imar planlarındaki şartlara aksi halde bu Yönetmelikte gösterilen yüksekliğe veya kat adedine uyulması mecburidir.	Applicability con.
13	Belirlenen gabari içinde kalmak şartıyla iç yükseklikleri arttırarak daha az adette kat yapılabilir.	Rule
14	Ayrıca; yapı emsali verilmemiş adalarda, İmar Kanunu, imar planı ve bu Yönetmeliğe göre çok katlı yapı yapılması mümkün olan parsellerde, sahibinin talebi halinde, *encümence uygun görülecek muvakkat bir zaman için* yüksekliği 6.80 m.yi ve 2 katı geçmeyen yapı ve tesislere tespit edilen kitle nizamına uygun olmak koşulu ile *encümence izin verilebilir*.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C30	ST2	R30.1	Self-contained	Formalizable
	ST3	R30.2	Linked-explanatory	Formalizable
	ST4	R30.3	Linked-explanatory	Formalizable
	ST5	R30.4	Linked-explanatory	Formalizable
	ST6	R30.5	Linked-explanatory	Formalizable
	ST7	R30.6	Linked-explanatory	Formalizable
	ST8	R30.7	Linked-explanatory	Formalizable
	ST9	R30.8	Linked-explanatory	Formalizable
	ST10	R30.9	Linked-explanatory	Formalizable
	ST11	R30.10	Linked-explanatory	Formalizable
	ST13	R30.11	Self-contained	Semi-formalizable
	ST14	R30.12	Self-contained	Non-formalizable

	Textual Expressions of Clause 38	Statement Type
	Madde 38– Zemin Kat Döşeme Seviyeleri	*Id&heading*
1	Zemin kat döşeme seviyeleri, binaların kot aldığı yol cephesince bu kota esas olan tretuvar üst seviyesinden itibaren + 0.50m. ile +1.00 m. arasında düzenlenir.	Rule
2	Binaların zemin katlarının işyeri veya otopark olarak kullanılması halinde, +0.50 m. koşulu aranmaz, bu durumda, zemin kat döşeme seviyeleri tretuvar üst seviyesinden aşağı indirilemez.	Rule
3	Ayrıca, fazla meyilli sokaklar üzerinde yapılacak dükkan, otopark ve benzeri girişlerin tretuvar kotlarına uydurulması amacıyla yapılacak döşeme kademeleri yukarıdaki sınırlamalar dışındadır.	Clarification
4	Arazi doğal zemin kotlarına uymak amacıyla veya *mimari gereksinmeler nedeniyle*, bir bina bloğunun, bir binanın veya müstakil bir bağımsız bölümün tespit edilen bina yüksekliğini aşamamak, belirli piyesler için tayin olunan asgari kat yüksekliklerine ve bu Yönetmeliğin diğer hükümlerine aykırı olmamak şartı ile çeşitli kotlardan ve farklı taban veya tavan seviyelerinden müteşekkil olarak tertiplenmesi mümkündür.	Rule
5	Ancak, bu durumda zemin katın en yüksek döşeme kotu, kot alma noktasından itibaren +1.50 m. yi geçemez.	Rule
6	Bu kademelendirme kitle hattından itibaren yol cephesinden 3.00 m. geriden başlayarak bütün bina derinliğince ve diğer cepleri boyunca da yapılabilir.	Clarification
7	Ancak yol cephelerinde imar planında belirtilen kat adedi aşılamayacağından sadece tek yola cepheli parsellerde uygulanır.	Clarifiaction
8	Fazla meyilli yollarda köşe başı olmayan parsellerde, yol cephesinde yolun meylinden dolayı zemin kat taban kotunun tretuvardan en fazla 3.00 m. yükseldiği noktalarda binada *kademe* yaptırılır.	Rule
9	Yol cephesinde en düşük son kademe, cephe boyunca 6.00 m.den aşağı olamaz.	Rule
10	6.00m.den az olması durumunda bir önceki kademe seviyesine uyulur.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C38	ST1		Self-contained	Formalizable
	ST2		Linked-explanatory	Fromalizable
	ST4		Self-contained	Non-formalizable
	ST5		Linked-explanatory	Formalizable
	ST8		Self-contained	Semi-formalizable
	ST9		Self-contained	Formalizable
	ST10		Linked-explanatory	Formalizable

	Textual Expressions of Clause 39	Statement Type
	Madde 39– Bazı Yapılarda Aranan Şartlar	*Id&heading*
1	Ahşap ve yarı ahşap binalar bitişik yapılamazlar.	Rule
2	Ayrık nizamda ise komşu sınırlarına bırakılacak min. bahçe mesafesi 5.00 m.den az olamaz.	Rule
3	Yüksekliği max 6.80 m.yi aşamaz.	Rule
4	Ahşap ve yarı kagir binaların komşu hududuna zeminden itibaren çatının her yerinde 0.60 m.lik yüksekliğe kadar ve en az bir tuğla kalınlığında yangın duvarı yapılması koşuluyla bitişik olarak inşa edilebilir.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
39	ST1		Self-contained	Formalizable
	ST2		Self-contained	Formalizable
	ST3		Self-contained	Formalizable
	ST4		Self-contained	Formalizable

	Textual Expressions of Clause 40	Statement Type
	Madde 40– Saçak ve Güneş Kesiciler	*Id&heading*
1	Binalarda son kat tavan döşemelerinde binanın çıkma hattından itibaren 0.50m. yi geçmemek üzere tüm bina cepheleri boyunca saçak yapılabilir.	Rule
2	Ön bahçesiz nizama tabi parsellerde bina cephesinden itibaren genişliği 1.50 m.yi geçmemek, konsol olmak ve komşu parsellere 2.00m. den fazla yaklaşmamak koşulu ile bina giriş ve dükkan önü saçakları düzenlenebilir.	Rule
3	Ancak bu saçakların en alçak noktası tretuvar üst seviyesinden en az 3.00 m. yükseklikte olabilir ve genişliği tretuvar genişliğini aşamaz.	Rule
4	Ön bahçeli nizama tabi parsellerde ise, tretuvara taşmamak, *civara ve binanın karakterine uygun olmak*, ve konsol olmak koşulu ile giriş saçakları düzenlenebilir.	Rule
5	Bina cephelerinde *alan olarak yararlanmamak* koşulu ile 1.00 m.yi geçmeyen güneş kesiciler yapılabilir.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C40	ST1	R40.1	Self-contained	Formalizable
	ST2	R40.2	Linked-explanatory	Formalizable
	ST3	R40.3	Linked-explanatory	Formalizable
	ST4	R40.4	Linked-explanatory	Formalizable**
	ST5	R40.5	Self-contained	Formalizable**

	Textual Expressions of Clause 41	Statement Type
	Madde 41– Çatılar	*Id&heading*
1	Çatılar binanın cephe aldığı yolun *yapı karakterine ve yapıya uygun nitelikte* olmalıdır.	Rule
2	Genel olarak çatıların %33 meyilli gabari dahilinde kalması şarttır.	Rule
3	Ancak, ayrık yapı nizamına tabi 2 katı geçmeyen dubleks konut yapılarında, imar durumunda belirtilen gabariye göre %33 meyil hesaplanarak bulunan mahya kotu aşılmamak kaydıyla, çatı eğimi ve çatı biçimi serbesttir.	Rule
4	Mahya kotu, mahya izdüşümü bina kitlesinin ½ sinden fazla olmamak kaydıyla beşik çatıya göre hesaplanır.	Clarification
5	Çatı eğimi , saçak genişliği dikkate alınmadan binanın cephesinden hesaplanır.	Clarification
6	Ancak, kapalı çıkma bulunan ve bu çıkma bina yüksekliğince devam eden binalarda çatı eğimi çıkma ucundan hesaplanır.	Clarification
7	Dubleks konut yapıları dışında kalan yapılarda, çatı yapılması halinde, mahya yüksekliği 3.00 m. yi aşmamak kaydıyla her cepheye akıntılı çatı yapılacaktır.	Rule
8	Çatı aralarına bağımsız bölüm yapılamaz.	Rule
9	Bu kısımlarda ancak, asansör kulesi, merkezi klima tesisatı, baca ve 6.80 m. ve daha az yükseklikteki dubleks konut binalarında içeriden irtibatlandırılmak, ait olduğu bağımsız bölüm sınırlarını aşmamak ve bu bağımsız bölümün son kattaki alanının %30'unu geçmemek kaydıyla piyesler yapılabilir.	Rule
10	Bu piyeslerden iskan edilenlerinde yükseklik en düşük yerde 1.80 m.den az olamaz.	Rule
11	Çatı arasının yukarıdaki şekilde düzenlenmesi halinde piyes önleri teras olarak tertiplenemez.	Rule
12	Çatı arasının son kat bağımsız bölümü ile birlikte kullanılması amacıyla son kat tavan betonu *kısmen veya tamamen* yapılmayabilir.	Rule
13	Son kat tavan döşemesi en yüksek mahya kotunu aşmayacak ve en fazla çatı eğimi içinde kalacak şekilde eğimli olarak tertip edilebilir.	Rule
14	Işıklıklar, alın ve kalkan duvarları, güneş enerjisi panelleri ve depoları çatı sathını 0.60 m.den fazla geçemez.	Rule
15	Duman ve hava bacaları hariç çatı üzerine hiçbir çıkma ve çıkıntı yapılamaz.	Rule
16	Ancak, 2.10 m. iç yüksekliği aşmamak kaydıyla düzenlenen merdiven evleri ve *Türk Standartları şartlarının gerektirdiği hallerde* asansör kulelerinin çatı örtüsünü aşmasına izin	Rule
17	Teras çatılarda asansör kulesi, merkezi anten, 2.10 m. iç yüksekliği aşmamak kaydıyla düzenlenen merdiven evi ve 1.10 m. parapet yüksekliğini aşmamak kaydıyla güneş enerjisi, su deposu, merkezi klima gibi tesisler yapılabilir.	Rule
18	Teras çatılarda en çok 1.10 m. yükseklikte yapılacak kagir korkuluk bina yüksekliğine dahil edilmez.	Clarification

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C41	ST1	R41.1	Self-contained	Non-formalizable
	ST2	R41.2	Self-contained	Formalizable
	ST3	R41.3	Linked-explanatory	Formalizable
	ST7	R41.4	Self-contained	Formalizable
	ST8	R41.5	Self-contained	Formalizable
	ST9	R41.6	Linked-explanatory	Formalizable
	ST10	R41.7	Linked-explanatory	Formalizable
	ST11	R41.8	Linked-explanatory	Formalizable
	ST12	R41.9	Linked-explanatory	Semi-formalizable
	ST13	R41.10	Linked-explanatory	Formalizable
	ST14	R41.11	Self-contained	Formalizable
	ST15	R41.12	Self-contained	Formalizable
	ST16	R41.13	Linked-explanatory	Formalizable**
	ST17	R41.14	Self-contained	Formalizable

Textual Expressions of Clause 42		Statement Type
Madde 42– Çıkmalar		*Id&heading*
1	Binalarda taban alanı (azami bina sahası) dışında kendi bahçe hudutları dışına taşmamak ve genişliği 1.50 m.yi aşmamak, tabii veya tesviye edilmiş (37/c maddesine göre) zeminden çıkma altına kadar en yakın şakuli mesafe 2.40m.nin altına düşmemek kaydıyla;	Rule
2	Parselin yan ve arka komşu hududuna 3.00 m.den fazla yaklaşmamak şartı ile açık ve kapalı çıkma yapılabilir ve bu çıkma cephe uzunluğunca devam edebilir.	
3	Yan bahçe mesafesi 3.00m. ile 4.00m. arasında olan parsellerde, yan komşu hududuna 2.00m. den fazla yaklaşmamak kaydıyla 1.00 m. ye kadar açık çıkma yapılabilir, ancak bu çıkma binanın çıkma yapılan cephe uzunluğunun 1/3'inden fazla olamaz.	Rule
4	Bitişik yapı nizamına tabi binaların ön ve arka cephelerinde yapılacak çıkmalar yan komşu hududuna 2.00 m.ye kadar yaklaştırılabilir.	Rule
5	Ayrıca, bitişik nizama tabi yerlerde, iki taraftaki *ilgililerin muvafakatı halinde ve ilgili belediyece mahzur görülmediği takdirde*, çıkmaların yan komşu hududuna yaklaştırılmalarına da izin verilebileceği gibi, komşu parselde *imar planına göre aynen muhafazası gereken* ve arka cephe hattı çıkma yapılacak binanınkinden ileride olan bir bina bulunması halinde çıkmanın bu hattı aşmamak üzere bu tarafta komşu hududuna kadar devam ettirilmesi de mümkündür.	Rule
6	Bitişiğinde çıkması komşu parsel hudutlarına kadar dayanmış ruhsatlı ve gabarisinde teşekkül etmiş mevcut bir bina var ise, muvafakat aranmaksızın çıkmanın komşu hududa kadar yanaşmasına müsaade edilir.	Rule
7	Yol genişliği (6.00) m.den büyük ön bahçesiz parsellerde; (yol genişliği - 6.00 m.) / 2 formülüyle hesaplanarak açık ve kapalı çıkma yapılmasına izin verilir.	Rule
8	Ancak, çıkma genişliği her durumda 1.00 m. den fazla olamaz ve yol projesine göre tespit edilen tretuvar üst kotundan çıkma altına kadar olan şakuli mesafe (3.00) m.den az olamaz.	Rule
9	İki yanında imar planına, mevzuata ruhsat ve eklerine uygun olarak teşekkül etmiş ve formüle göre bulunacak değerden daha geniş çıkmalı binalar olması halinde yeni yapılacak binaya da bitişik binalarla *uyum sağlayacak şekilde* çıkma izni verilir.	Rule
10	Zemin katta kendi parsel hududu dışına taşmayan, hangi katta yapılırsa yapılsın (0.20) m.yi geçmeyen motif çıkmalar yapılabilir.	Rule
11	Parselin bulunduğu ada yüzünün tamamının yeşil sahaya veya açık otoparka bakması halinde, yol genişliğine bakılmaksızın max. (1.00) m. ye kadar açık ve kapalı çıkma izni verilir.	Rule
12	Parselin bulunduğu ada yüzünün karşı hattının kısmen yeşil saha kısmen imar adası olması halinde, yol genişliği dikkate alınarak yukarıdaki formüle göre çıkma izni verilir.	Clarification
13	Yol genişliğinin tayininde yollardaki arızi (devamlılık arz etmeyen) genişleme ve daralmalar dikkate alınmaz.	Clarification
14	Uygulama, ada boyutunda muhtelif noktalardan alınacak yol genişliklerinin ortalama değerine göre yapılır.	Clarification

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C42	ST1	R42.1	Self-contained	Formalizable
	ST2	R42.2	Linked-explanatory	Formalizable
	ST3	R42.3	Linked-explanatory	Formalizable
	ST4	R42.4	Linked-explanatory	Formalizable
	ST5	R42.5	Linked-explanatory	Non-formalizable
	ST6	R42.6	Linked-explanatory	Formalizable
	ST7	R42.7	Linked-explanatory	Formalizable
	ST8	R42.8	Linked-explanatory	Formalizable
	ST9	R42.9	Linked-explanatory	Non-formalizable
	ST10	R42.10	Self-contained	Formalizable
	ST11	R42.11	Linked-explanatory	Formalizable

	Textual Expressions of Clause 43	Statement Type
	Madde 43–Işıklıklar ve Hava Bacaları *Id&heading*	
	A- Konut Yapılarında: *subheading*	
1	Her müstakil ev veya dairede, en az 1 oturma odası ile yatak odalarının doğrudan doğruya hariçten ışık ve hava almaları gereklidir.	Rule
2	Bu şekilde ışık ve hava almalarına lüzum olmayan diğer odalarla mutfakların ışıklıktan, yıkanma yeri ve helaların ışıklık veya hava bacasından faydalanmaları da mümkündür.	Rule
3	Işıklıklar 1 ve 2 katlı binalarda; dar kenarı 1.00 m.den, alanı 3.00 m2'den az olamaz.	Rule
4	Işıklıklar 3, 4 ve 5 katlı binalarda; dar kenarı 1.50 m.den, alanı 4.50 m2'den az olamaz.	Rule
5	Işıklıklar 6,7,8,9 katlı binalarda, dar kenarı 2.00 m.den, alanı 6.00 m2'den az olamaz.	Rule
6	Işıklıklar 10 ve daha fazla katlı binalarda; dar kenarı 2.00 m.den, alanı 9.00 m2'den az olamaz.	Rule
	B- Konut Dışı Yapılarda: *subheading*	
7	Otel, pansiyon, iş hanı ve benzeri binalarda, odalar *gereğinde* ışıklığa açılabilir.	Rule
8	Işıklıklar 1 ve 2 katlı binalarda; dar kenarı 1.50 m.den, alanı 4.50 m2'den az olamaz.	Rule
9	Işıklıklar 3, 4,5, 6,7,8,9 katlı binalarda; dar kenarı 2.00 m.den, alanı 6.00 m2'den az olamaz.	Rule
10	Işıklıklar 10 ve daha fazla katlı binalarda; dar kenarı 2.00 m.den, alanı 9.00 m2'den az olamaz.	Rule
	C- Genel Hükümler: *subheading*	
11	Her türlü binada sadece havalandırma amacı ile kullanılan hava bacalarının asgari ölçüsü (0.60 x 0.60) m2. , içinden tesisat geçirilen hava bacalarının asgari ölçüsü ise (0.80 x 0.80) m2. olup, bu alan herhangi bir yapı elemanı ile (baca, kiriş, vs.) daraltılamaz.	Rule
12	Asgari ölçüdeki bir ışıklık veya hava bacasından her katta en çok 4 piyes faydalanabilir.	Clarification
13	Bu piyeslerin adetlerinin artması halinde, 4'ten fazla her piyes için ışıklık veya hava bacası ölçüsü *aynı nispette* artırılır.	Rule
14	Ancak, yukarıda belirtilen şekilde ışık ve hava alması gerekmeyen veya lüzumlu ışık ve havayı yönetmelikte tarif edilen şekilde esasen alması mümkün olan piyeslerden, herhangi bir ışıklık veya hava bacasına pencere açılması, bu ışıklık veya hava bacası ölçülerinin artırılmasını gerektirmez.	Clarification
15	Her binanın *lüzumlu* ışıklık veya hava bacası, kendi parseli üzerinde bulunacaktır.	Rule
16	Komşu bina ve parselin ışıklık veya hava bacasından faydalanmak suretiyle, bu elemanların yapılmamasına veya ölçülerinin azaltılmasına izin verilmez.	Clarification
17	Işıklık veya hava bacaları bunlara ihtiyacı olan kattan itibaren başlatılabilir.	Clarification
18	Ancak meyilden kat kazanılması halinde yapının toplam kat sayısına tekabül eden yüksekliğe ait, ışıklık ölçüleri uygulanacaktır.	Clarification
19	Bu arada ışıklık veya hava bacalarının bunlara ihtiyacı olan kattan itibaren başlatılması halinde yapının ışıklığının başladığı kat sayısına tekabül eden yüksekliğe ait ışıklık ölçüleri uygulanır.	Clarification
20	Ayrıca asansör, merdiven gibi kısımlar aydınlık alanına *tecavüz edemez*.	Rule
21	Işıklıkların içerisine açık renk boya ve badana yapılması mecburidir.	Out of Scope
22	Bina altında 2.50 m.den geniş geçitlerde ışıklık olarak kullanılabilir.	Clarification
23	Kafeterya, restoran, kahvehane, lokanta, atölye, imalathane, diskotek gibi insanların toplu olarak bulunduğu mekanlarda doğal havalandırma dışında, mekanik havalandırma yapılması zorunludur.	Out of Scope

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C43	ST1	R43.1	Self-contained	Formalizable
	ST2	R43.2	Linked-explanatory	Formalizable
	ST3	R43.3	Self-contained	Formalizable
	ST4	R43.4	Linked-explanatory	Formalizable
	ST5	R43.5	Linked-explanatory	Formalizable
	ST6	R43.6	Linked-explanatory	Formalizable
	ST7	R43.7	Self-contained	Semi-formalizable
	ST8	R43.8	Self-contained	Formalizable
	ST9	R43.9	Linked-explanatory	Formalizable
	ST10	R43.10	Linked-explanatory	Formalizable
	ST11	R43.11	Self-contained	Formalizable
	ST13	R43.12	Linked-explanatory	Semi-formalizable
	ST15	R43.13	Self-contained	Semi-formalizable
	ST20	R43.19	Self-contained	Semi-formalizable

Textual Expressions of Clause 44		Statement Type
Madde 44–Yapılarda Bulunmasi Gereken Piyesler ve Ölçüler		*Id&heading*
1	Her bağımsız konutta en az bir yaşam mekanı, bir yatak odası veya nişi, bir mutfak veya yemek pişirme nişi, bir banyo (WC ile birlikte) veya bir yıkanma yeri ile bir WC bulunması zorunludur.	Rule
2	Bu mekanlar aşağıda belirtilen ölçülerden küçük yapılamaz.	Clarification
3	Yaşam mekanı; dar kenarı 3.00 m.den, alanı 12.00 m2,	Rule
4	Yatak odası; dar kenarı 2.60 m, alanı 7.28 m2,	Rule
5	Yatak nişi; dar kenarı 1.50 m, alanı 3.00 m2,	Rule
6	Mutfak; dar kenarı 1.50 m, alanı 3.60 m2,	Rule
7	Yemek pişirme yeri; dar kenarı 0.70 m, alanı 1.40 m2,	Rule
8	Banyo (WC ile birlikte); dar kenarı 1.20 m.den, alanı 3.48 m2,	Rule
9	Yıkanma yeri; dar kenarı 1.20 m, alanı 2.64 m2,	Rule
10	WC; dar kenarı 0.90m, alanı 1.08 m2,	Rule
11	antre, hol ve benzeri geçitler; dar kenarı 1.00 m, alanı 1.32 m2,	Rule
12	Birden fazla daire ile ilgili genel geçitler; dar kenarı 1.10 m, alanı 1.32 m2,	Rule
13	Tabloda belirtilen mekanlar dışında ayrılmak istenen çalışma odası, hobi odası gibi kullanımlara ilişkin mekanların dar kenarı 2.10 m.den ve alanı 6.00 m2.den az olamaz.	Rule
14	Yatak nişleri dar kenarı (3.00) m.den ve alanı (12.00) m2 olan bir yaşam mekanı açılacaktır.	Rule
15	Yemek pişirme yerleri hava ve duman bacaları ile irtibatlı olmak şartı ile düzenlenebilir.	Rule
16	Sobalı ısıtma sistemi seçilen yapılarda ayrıca en az 2.50 m2 net alanlı kömürlük (odunluk) ayrılacaktır.	Rule
17	Bu hacim binanın bodrum katında veya müştemilat bölümünde de yapılabilir, ancak daire içinde yapılması halinde max. (4.50) m2'yi geçemez.	Rule
18	Islak hacimlerde tefriş yapılması zorunludur.	Rule
19	Umumi binalarda koridor genişlikleri; uzunluğu (20.00) m.ye kadar olan koridorlar (2.00)m.den, (20.00) m.yi geçen koridorlar (2.50) m.den dar olamaz.	Rule
20	İmar planlarında aksine bir açıklama olmaması halinde, her türlü işyerinin cephesi 2.00 m. den az olamaz.	Rule
21	Resmi kurumlarca yaptırılacak; eğitim binaları, sağlık binaları, spor tesisleri ve bu gibi binaların da ilgili bakanlıklarca onaylanmış projeleri esas alınır.	Clarification
22	Bu amaçla yaptırılacak özel binalarda, bu Yönetmelikte belirtilen koşullar dışında ilgili bakanlıkların yönetmelikleri dikkate alınacaktır.	Clarification

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C44	ST1	R44.1	Self-contained	Formalizable
	ST3	R44.2	Self-contained	Formalizable
	ST4	R44.3	Self-contained	Formalizable
	ST5	R44.4	Self-contained	Formalizable
	ST6	R44.5	Self-contained	Formalizable
	ST7	R44.6	Self-contained	Formalizable
	ST8	R44.7	Self-contained	Formalizable
	ST9	R44.8	Self-contained	Formalizable
	ST10	R44.9	Self-contained	Formalizable
	ST11	R44.10	Self-contained	Formalizable
	ST12	R44.11	Self-contained	Formalizable
	ST13	R44.12	Self-contained	Formalizable
	ST14	R44.13	Self-contained	Formalizable
	ST15	R44.14	Self-contained	Formalizable
	ST16	R44.15	Self-contained	Formalizable
	ST17	R44.16	Linked-explanatory	Formalizable
	ST18	R44.17	Self-contained	Formalizable
	ST19	R44.18	Linked-explanatory	Formalizable
	ST20	R44.19	Self-contained	Formalizable

Textual Expressions of Clause 45		Statement Type
Madde 45–İç Yükseklikler	*Id&heading*	
1	Genel olarak iskan edilen katların taban döşeme kaplaması üzerinden tavan altına kadar olan net (döşeme kaplamaları ve sıvalar ikmal edildikten sonra) yüksekliği 2.60m.den az olamaz.	Rule
2	Projelerde döşeme ve tavan kaplama detayı gösterilmediği takdirde bu yüksekliğe sıva ve kaplama payı olarak (0.07) m. ilave edilir.	Clarification
3	Yıkanma yeri, banyo, duş, lavabo yeri, WC, kiler, ofis, antre, koridor, yatak holü, merdiven altı, her türlü iç ve dış geçitler, iskan edilemeyen bodrum katları ile müştemilat binalarında, bu yükseklik net (2.20) m.den aşağıya düşmemek üzere indirilebilir.	Rule
4	Garaj ve otoparkların yükseklikleri kiriş altı net 2.00 m. den az olamaz.	Rule
5	Otel, pansiyon, iş hanı, büro ile benzeri işyerleri ve içerisinde insan oturan, yatan veya çalışan diğer binaların iç yükseklikleri döşeme kaplaması ve tavan sıvası hariç (2.60) m.den daha az olamaz.	Rule
6	Mağaza ve dükkanlar ile pastaneler, içkili ve içkisiz lokantalar vb yemek yerleri, kahvehanelerin taban döşemesi üzerinden tavan altına kadar net yüksekliği (3.00) m.den az olamaz.	Rule
7	Düğün ve oyun salonları, diskotek, birahaneler ve gazino mahallerinin taban döşemesi üzerinden tavan altına kadar net yüksekliği (3.50) m.den az olamaz.	Rule
8	Müştemilat ve servis kısımlarında net yükseklik (2.60) m.den az olamaz.	Rule
9	Ve bu bölümler toplam alanın %50'sini geçemez.	Rule
10	Resmi binalarla ilgili bakanlıkça onaylanmış projeler esas alınacaktır.	Clarification
11	Bu amaçla yaptırılacak özel binalarda bu Yönetmelikte belirtilen koşullar dışında, ilgili bakanlıkların yönetmelikleri dikkate alınacaktır.	Clarification

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C45	ST1	R45.1	Self-contained	Formalizable
	ST3	R45.2	Linked-explanatory	Formalizable
	ST4	R45.3	Linked-explanatory	Formalizable
	ST5	R45.4	Linked-explanatory	Formalizable
	ST6	R45.5	Linked-explanatory	Formalizable
	ST7	R45.6	Linked-explanatory	Formalizable
	ST8	R45.7	Linked-explanatory	Formalizable
	ST9	R45.8	Linked-explanatory	Formalizable

Textual Expressions of Clause 46		Statement Type
Madde 46–Pencereler	*Id&heading*	
1	Binaların pencere boşlukları dar kenarı (0.60) m.den az olmamak şartı ile, toplam faydalanılacak piyes alanının yaşam mekanı, oda ve mutfakların da 1/8'inden ve her durumda (1.25) m2'den az olamaz.	Rule
2	Camlı balkon kapılarında pencere boşluğu sayılır.	Clarification
3	Dubleks konut yapılarında çatı arasına yapılan mekanların pencere boşlukları (0.80)m2 den büyük olmamak ve her mekana çatı üzerinde en çok iki pencere açılmak ve pencereler birbirine eklenerek bant haline getirilmemek şartı ile yapılabilir.	Rule
4	Isı yalıtım yönetmeliği hükümleri saklıdır.	Clarification
5	Binaların bitişik komşu tarafına; ilgili komşu parsel sahibinin muvafakati alınıp tapuya tescil ettirilmedikçe pencere ve kapı açılamaz.	Rule
6	Pencere veya kapı açılacak bu piyesin, ayrıca, gerekli ışık ve havayı doğrudan alacak elemanlara haiz olması gerekir.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C46	ST1	R46.1	Self-contained	Formalizable
	ST3	R46.2	Linked-explanatory	Formalizable
	ST5	R46.4	Self-contained	Non-formalizable
	ST6	R46.5	Linked-explanatory	Non-formalizable

Textual Expressions of Clause 47		Statement Type
Madde 47–Kapılar	*Id&heading*	
1	Kapı Yükseklikleri: Kasa dahil (2.10) m.den az olamaz.	Rule
Kapı Genişlikleri:	*subheading*	
2	Birden fazla bağımsız bölümü olan binaların ana giriş kapıları kasa dahil (1.30) m.den,	Rule
3	Bağımsız bölüm kapıları, kasa dahil (1.00) m.den,	Rule
4	Oda ve mutfak kapıları kasa dahil (0.90) m.den,	Rule
5	Yıkanma yeri, WC, odunluk, kömürlük, kiler kapıları kasa dahil (0.80) m.den	Rule
6	Dükkan kapıları, kasa dahil (1.00) m.den, az olamaz.	Rule
7	Asansör, garaj ve benzeri özellik arz eden yerlerin kapı boyutları hizmetin gerektirdiği şekilde tespit edilir.	Rule
8	Banyo kapılarında, alttan temiz hava girecek şekilde bir düzen bulunacaktır.	Rule
9	Umumi binalarda, bina ana giriş kapısına merdivenle ulaşılıyorsa bedensel özürlülerin kullanımı için en fazla %6 eğimli, en az (1.20) m. genişlikte koruma bordürlü ve korkuluklu rampa yapılacaktır.	Out of Scope
10	İç kapılar tamamen eşiksiz ve en az (0.95) m. genişliğinde olacaktır.	Out of Scope
11	Umumi binalarda, bütün kapılar kaçış yönüne açılacaktır.	Out of Scope

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C47	ST1	R47.1	Self-contained	Formalizable
	ST2	R47.2	Linked-explanatory	Formalizable
	ST3	R47.3	Linked-explanatory	Formalizable
	ST4	R47.4	Linked-explanatory	Formalizable
	ST5	R47.5	Linked-explanatory	Formalizable
	ST6	R47.6	Linked-explanatory	Formalizable
	ST7	R47.7	Linked-explanatory	Non-formalizable
	ST8	R47.8	Self-contained	Formalizable

	Textual Expressions of Clause 48	Statement Type
Madde 48–Asansörler	*Id&heading*	
1	Bina giriş katından itibaren yüksekliği 12.80 m.yi geçen ve 4'ten fazla katı bulunan konut yapıları ile, yüksekliği 6.80 m.yi geçen ve 2'den fazla katı bulunan konut dışı yapılarda, giriş katından itibaren (çatı katı hariç) son kata kadar ve varsa iskan edilebilir bodrum katlara da inmek koşuluyla kullanılan tüm katlara hizmet verecek şekilde, yürürlükteki Türk Standartları Enstitüsünün standartları ve Asansör Yönetmeliğine uygun asansör tesisi zorunludur.	Rule
2	Bina giriş kat kotundan son kat kotuna kadar olan yükseklik veya bu yükseklik içindeki kat sayısı bu maddenin uygulanmasında asansör yapılması mecburiyetine esas alınacaktır.	Applicability con.
3	Yüksekliği 12.80 m'yi geçen ve girişten itibaren daire adeti 20 den fazla olan meskenlerde (her iki şartın bir arada gerçekleşmesi halinde) çift asansör yapılması mecburidir.	Rule
4	Yüksekliği 12.80 m'yi geçen ve kat alanı 250 m2 den fazla olan ticari amaçlı (büro, işhanı, çarşı, benzeri) yapılarda, her iki şartın bir arada gerçekleşmesi halinde, *Elektrik ve Makine Mühendisleri Odası Asansör Avan ve Uygulama Projeleri Hazırlama ve Teknik Esaslarında yer alan trafik hesabı yapılarak* asansör sayısı tespit edilecektir.	Rule
5	Binanın kat ve daire adedinin fazlalığı veya kullanma şeklinin gerektirdiği *lüzuma göre*, asansör ve yerinin ölçü veya adedini arttırmaya başlangıç katı olarak zemin kat yerine bodrum veya birinci katı *kabul veya tayine belediye yetkilidir*.	Rule
6	İmar planı ile kanun, tüzük ve yönetmelik hükümlerine göre *muhafazası mümkün olan* binalarda kat ilavesi halinde, ilave kat ile birlikte kat adedi beşi ve bina yüksekliği 15.80 m.yi geçmediği taktirde, *asansör aranmayabilir veya asansör yeri ölçüleri mevcuda uydurulabilir.*	Rule
7	Binalarda usulüne göre asansör yapılmış olması nizami şekil ve ölçülerle merdiven yapılması şartını ortadan kaldırmaz.	Clarification
8	Asansör makine daireleri yürürlükteki *Türk Standartları Enstitüsü standartlarının gerektirdiği* minimum ölçülerde düzenlenebilir.	Rule
9	*Teknik koşulların gerektirdiği durumlarda* bu alan % 30 kadar arttırılabilir.	Rule
10	Asansörün yapılması ve işletilmesi ile ilgili hususlarda yukarıdaki hükümlerde dikkate alınarak yürürlükteki Asansör Yönetmeliği ve Türk Standartları Enstitüsü standartları hükümlerine uyulur.	Clarification

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C48	ST1	R47.1	Self-contained	Formalizable
	ST3	R47.2	Linked-explanatory	Formalizable
	ST4	R47.3	Linked-explanatory	Non-formalizable
	ST5	R47.4	Linked-explanatory	Non-formalizable
	ST6	R47.5	Linked-explanatory	Non-formalizable
	ST8	R47.6	Self-contained	Non-formalizable
	ST9	R47.7	Linked-explanatory	Non-formalizable

	Textual Expressions of Clause 49	Statement Type
	Madde 49–Merdivenler	*Id&heading*
1	Bu yönetmelikte sözü geçen umumi binalarla, otel, işhanı, büro, pasaj, çarşı ve benzerlerinin, birden fazla katı olan ev ve apartmanların ahşap olmayan en az bir ana merdiveni olacaktır.	Rule
	A- Genel Hükümler:	*subheading*
2	Merdivenlerin minimum kova ölçüsü 20 cm. olmalıdır.	Rule
3	Merdiven evlerinin bina cephesinden, çatıdan veya ışıklıktan doğrudan ışık ve hava alması gereklidir.	Rule
4	Merdivenlerin çatı, bodrum ve benzeri ortak alanlara ulaştırılması zorunludur.	Rule
5	Bu durumda merdiven kolu ve sahanlık genişliği 1.20 m.den az olamaz.	Rule
6	Merdiven basamaklarının ölçüleri; 2a + b = 60 ile 63 formülüne göre hesaplanır.	Rule
7	Formüldeki (a) basamak yüksekliğini, (b) basamak genişliğini gösterir.	Clarification
8	Merdiven tanziminde her 18 rıhtan sonra ara sahanlık bırakılacaktır.	Rule
9	Bütün binalarda kat ve ara sahanlıkların genişliği, merdiven kolu genişliğinden az olamaz.	Rule
10	Düz kollu veya döner merdivenlerde çıkış hattında yapılan ara sahanlıklar 1.00 m. olabilir.	Rule
11	İmar Kanunu ile imar planı ve bu Yönetmelik hükümlerine göre korunması mümkün olan binalarda kat ilavesi halinde, mevcut merdivenler bu madde hükümlerine uymadığı takdirde, bu konuda yapılacak işlemi saptamaya ilgili belediyesi yetkilidir	Rule
	B-Konut Yapılarında	*subheading*
12	Merdiven kolu ve sahanlık genişlikleri; 1.20 m.den az olamaz.	Rule
13	Bu ölçüler tek aileye mahsus evlerde ve bodrum katlarıyla, servis merdivenlerinde (0.90) m.ye kadar indirilebilir.	Rule
14	İskan edilmeyen çatı aralarına çıkan merdivenlerde bu ölçüler aranmayabilir.	Rule
15	Merdiven basamaklarının yüksekliği (0.175) m.den fazla, basamak genişliği ise çıkış hattında (0.28) m.den, kovada (0.10) m.den az olamaz.	Rule
	B-Konut Dışı Yapılarda	*subheading*
16	Merdiven kolu ve sahanlık genişliği; (1.60) m.den az olamaz.	Rule
17	Merdivenlerin her iki yanında korkuluk ve/veya küpeşte bulunmalıdır.	Rule
18	Merdiven Basamaklarının Ölçüleri; İç ve dış merdivenlerde rıht yüksekliği : (0.16) m.den fazla; İç ve dış merdivenlerde basamak genişliği: çıkış hattında (0.30) m.den, kovada (0.125) m.den az olamaz.	Rule
19	Bu yönetmelikte oluşturulması zorunlu tutulan ana merdivenler dışında düzenlenen merdivenlerde, umumi bina tanımına girmeyen yapılarda düzenlenen bağımsız bölüm içi merdivenlerde ve cephesi 8.00 m.den ve/veya kat alanı 150 m^2.den küçük binalarda düzenlenecek merdivenlerde; merdiven kolu ve sahanlık genişliği 1.20m.den az, basamak yüksekliği (0.175) m.den fazla, basamak genişliği ise, çıkış hattında (0.28) m.den, kovada (0.125) m.den az olamaz.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C49	ST1	R49.1	Self-contained	Formalizable
	ST2	R49.2	Self-contained	Formalizable
	ST3	R49.3	Self-contained	Formalizable
	ST4	R49.4	Self-contained	Formalizable
	ST5	R49.5	Linked-explanatory	Formalizable
	ST6	R49.6	Linked-explanatory	Formalizable
	ST8	R49.7	Self-contained	Formalizable
	ST9	R49.8	Linked-explanatory	Formalizable
	ST10	R49.9	Linked-explanatory	Formalizable
	ST11	R49.10	Self-contained	Non-formalizable
	ST12	R49.11	Linked-explanatory	Formalizable
	ST13	R49.12	Linked-explanatory	Formalizable
	ST14	R49.13	Linked-explanatory	Formalizable
	ST15	R49.14	Linked-explanatory	Formalizable
	ST16	R49.15	Linked-explanatory	Formalizable
	ST17	R49.16	Self-contained	Non-formalizable
	ST18	R49.17	Linked-explanatory	Formalizable
	ST19	R49.18	Linked-explanatory	Formalizable

Textual Expressions of Clause 50		Statement Type
Madde 50–Yangın Merdivenleri	*Id&heading*	
1	a) " Binaların Yangından Korunması Hakkında Yönetmelik" hükümlerine uyulacaktır.	Rule
2	Yangın merdiveni genişliği konut ve büro yapılarında min 90 cm. topluma açık diğer yapılarda min. 120 cm. olacaktır.	Rule
3	İrtifak hakkı tesisi suretiyle komşu parsellere ortak yangın merdivenleri düzenlenebilir.	Clarification
4	Ticaret bölgelerinde; bodrum ve zemin katları parselin tamamında inşaata müsaadeli binalarda, ana blok dışında, 5.50 m. kotundaki döşeme üzerine inilmesi ve *çıkış güvenliğini sağlaması*, genel merdivene ulaşan koridorun yangına dayanaklı olması, genel merdivenin ise yangın merdiveni özellikli olması koşulu ile yola irtibatı binanın genel hacimlerinden sağlanan yangın koridoru ve yangın merdiveni düzenlenebilir.	Rule
5	Kullanım amacı nedeniyle birden fazla merdiven yapılması gereken yapılarda, genel merdivenlerin ve yangın merdivenlerinin son kata kadar ulaşması ve bu katta birbirlerine karşı *güvenlikli olarak bağlanması* sağlanmalıdır.	Rule
6	Bu yönetmelikten önce yangın merdivensiz olarak yapılmış yapılarda; Binaların Yangından Korunması Hakkında Yönetmelik hükümleri saklı kalmak kaydıyla, ilave edilecek yangın merdivenlerinde, kat mülkiyeti yasasına uyulmak koşuluyla yapılacak yangın merdivenlerinin ölçü ve biçimi *İtfaiye Müdürlüğünün de görüşü alınarak binanın durumuna göre belirlenir.*	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C50	ST1	R50.1	Self-contained	Formalizable
	ST2	R50.2	Self-contained	Formalizable
	ST4	R50.3	Self-contained	Semi-formalizable
	ST5	R50.4	Self-contained	Semi-formalizable
	ST6	R50.5	Self-contained	Non-formalizable

Textual Expressions of Clause 51		Statement Type
Madde 51–Korkuluklar	*Id&heading*	
1	Her türlü binada balkon, kat ve çatı terasları etrafında, 5 ten fazla basamağı bulunan merdivenlerde, kotu (1.00) m. den az olan pencere boşluklarında, bina iç boşluklarında, döşeme kotundan itibaren en az (1.00) m. yüksekliğe kadar *fenni gereklere uygun olarak* korkuluk yapılması zorunludur.	Rule
2	Korkuluk araları, dikeyde, yatayda ve alt boşluklarında 10 cm. den fazla olamaz.	Rule
3	Korkuluklar yatay olarak düzenlendiği taktirde *merdiven etkisini önleyecek tedbirler* alınacaktır.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C51	ST1	R51.1	Self-contained	Formalizable
	ST2	R51.2	Self-contained	Formalizable
	ST3	R51.3	Self-contained	Semi-formalizable

	Textual Expressions of Clause 52		Statement Type
Madde 52–Bacalar		*Id&heading*	
A- Duman Bacaları:		*subheading*	
1	Yapılarda yer alması tasarlanan ısıtma sistemine göre, bacaların sayısı ve niteliği belirlenir.		Clarification
2	Bacaların proje tasarımlarında Türk standartları Enstitütüsü standartları, ısı yalıtım yönetmeliği ve doğal gaz iç tesisat yönetmeliğindeki şartlar sağlanacaktır.		Rule
3	Bacaların iç genişliği en az (0.13 x 0.13) m. olacaktır.		Rule
4	İki baca birbirine bağlanmayacağı gibi her ateş kaynağı ayrı bir bacaya bağlanacaktır.		Rule
5	Duman bacaları çatı örtüsünü en az 1.00 m, mahyayı ise en az 0.80 m. aşacaktır.		Rule
6	Çatı konstrüksiyonu ister ahşap, ister çelik olsun bacalara 0.10 m. den fazla yaklaşamaz.		Rule
7	Banyolara duman bacası yapılamaz.		Rule
a) Sobalı Isıtma Sistemi Seçilen Yapılarda;		*subheading*	
8	Yapıların konut olarak kullanılan her dairesinde mutfak ve mutfak nişi ile bunların dışında kalan en az iki piyesinde baca yapılması zorunludur.		Rule
9	Bu bacalar şönt baca olabilir, ancak, iki piyesten kömürlük dışındaki bir tanesinde yapılacak bacanın müstakil baca olması zorunludur.		Rule
10	Yapıların dükkan, mağaza v.b. kullanılan bölümlerinde her bağımsız bölüm için bir baca yapılması gereklidir.		Rule
b) Merkezi veya Kat Kaloriferi Isıtma Sistemi Seçilen Yapılarda;		*subheading*	
11	Her kalorifer kazanı için ayrı olarak düzenlenmiş müstakil bir duman bacası yapılması (şönt olamaz) zorunludur.		Rule
12	Ayrıca ; yapıların konut olarak kullanılan her dairesinde mutfak ve mutfak nişi ile bunların dışında kalan en az bir piyeste en az bir baca yapılması, otel, iş hanı, pasaj v.b umumi binaların ise her katında en az bir baca yapılması zorunludur.		Rule
13	Bu bacalar şönt olabilir.		Clarification
14	Kat kaloriferi kazanı balkona konulamaz, özel bir bölmeye konulduğunda bu mahallin en az 6 m3 hacminde olması, bina dış cephesinden, ışıklıktan veya hava bacasından havalandırılması ve müstakil bir duman bacasının bulunması zorunludur.		Rule
15	Kat kaloriferi kazanı ile yakıt tankı aynı mahale konulamaz.		Rule
c)Sınırları İlgili İdare Tarafından Belirlenecek Doğal Gaz Uygulama Bölgeleri İçinde İnşa Edilecek Yapılarda ;		*subheading*	
16	İskan edilebilir bodrum katlar dahil 5 katlı binaların mutfaklarında doğal gazla çalışan her cihaz için bir müstakil baca yapılacaktır.		Rule
17	5 kattan daha yüksek yapılarda (yüksek yapılar hariç) mutfakta doğal gaz için bir şönt baca yapılması yeterlidir.		Rule
18	10 katın üstündeki yapılarda; üstteki 10 kat şönt bacaya bağlanabilir, kalan diğer alt katlarda hermatik (denge bacalı) cihaz kullanılmalıdır.		Rule
19	Ayrıca; doğal gaz bacası dışında, kaloriferli yapılarda kalorifer bacası dışında düzenlenmesi ön görülen bütün bacalar yapılacaktır.		Rule
B-Tesisat Bacaları		*subheading*	
20	İçinden tesisat geçen bacaların en az (0.80x0.80) m. ebatlarında olması zorunludur.		Rule
21	Sadece tesisat amacı ile kullanılmak ve her katta ortak mahalle açılmak kaydıyla (0.40x0.40)m. ölçülerinde yapılabilir.		Rule
22	İçinden tesisat geçirilmeyen sadece havalandırma amacına yönelik hava bacaları ise (0.60x0.60)m. ölçülerinde yapılabilir.		Rule
23	Bacalar kiriş v.b herhangi bir yapı elemanı ile daraltılamaz.		Clarification
C-Çöp Bacaları		*subheading*	
24	Tüm binalarda istenildiğinde çöp bacaları yapılabilir.		Clarification
25	Zemin veya bodrum katlarında çöp toplama yerleri ve bağımsız bölüm bağlantılarının bulunması, iç yüzeylerinin pürüzsüz bir malzemeyle kaplanması ve kapak iç kısımlarının, hiçbir maddenin sızmasına olanak vermeyecek şekilde yapılması zorunludur.		Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C52	ST2		Self-contained	Formalizable
	ST3		Self-contained	Formalizable
	ST4		Self-contained	Formalizable
	ST5		Self-contained	Formalizable
	ST6		Self-contained	Formalizable
	ST7		Self-contained	Formalizable
	ST8		Self-contained	Formalizable
	ST9		Self-contained	Formalizable
	ST10		Self-contained	Formalizable
	ST11		Self-contained	Formalizable
	ST12		Linked-explanatory	Formalizable
	ST14		Self-contained	Formalizable
	ST15		Linked-explanatory	Formalizable
	ST16		Self-contained	Formalizable
	ST17		Linked-explanatory	Formalizable
	ST18		Linked-explanatory	Formalizable
	ST19		Linked-explanatory	Formalizable
	ST20		Self-contained	Formalizable
	ST21		Linked-explanatory	Formalizable
	ST22		Linked-explanatory	Formalizable
	ST25		Self-contained	Formalizable

Textual Expressions of Clause 53	Statement Type
Madde 53–Yangın Önlemleri	*Id&heading*
1 Bu yönetmelik esaslarına göre yapılacak tüm uygulamalarda, “ Binaların Yangından Korunması Hakkındaki Yönetmelik” hükümlerine uyulacaktır.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C53	ST1	R53.1	Self-contained	Formalizable

Textual Expressions of Clause 55		Statement Type
Madde 55–Bodrumlarla İlgili Hükümler		*Id&heading*
1	Binaların bodrum kısımları esas bloğa tabi değildir.	Clarification
2	Tabii zeminin veya tesviye edilmiş zeminin altında kalmak ve yol cephelerinde kitle hattına *tecavüz etmemek koşulu* ile bahçenin tamamında bodrum yapılabilir.	Clarification
3	Komşu parsellere *uyum sağlamak* ve İmar Kanunun 35. maddesindeki hükümler saklı kalmak kaydıyla, bu Yönetmeliğin 37/C maddesine uygun olarak, parsel tesviye edilebilir.	Clarification
4	Taban döşemesi üst seviyesi, tesviye edilmiş zeminin altında kaldığı takdirde mesken olarak kullanılamaz.	Rule
5	Ancak *imar planında belirlenen bölgeleme koşullarına uyulmak* ve bu Yönetmeliğin 43. maddesindeki şartları sağlamak kaydıyla işyerleri tesis edilir.	Rule
6	Konut bölgelerinde günlük ihtiyaçları karşılamaya dönük olarak zemin katların ticaret olarak planlanması halinde, bodrum katta aynı bağımsız bölümle içten bağlantılı piyesler tesis edilebilir.	Rule
7	Bu piyeslerin ayrı girişleri olamaz, binanın ortak alanları ve müştemilatlarıyla irtibatlandırılamaz.	Rule
8	Bodrum kapısı tamamen tretuvar üzerinde kalan fazla meyilli yollar dışında yapılacak ön bahçesiz binalarda yol cephesinde bodrum girişi yapılamaz.	Rule
9	Toprağa dayalı tüm bodrum katlar betonarme perde şeklinde inşa edilerek dış etkilere karşı ısı ve su yalıtımı yapılması zorunludur.	Rule
10	Ayrıca bu tür binalarda yer altı suyuna karşı gerekli drenaj yapılacak ve mimari proje ile mekanik tesisat projelerinde belirtilecektir.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C55	ST4		Self-contained	Formalizable
	ST5		Linked-explanatory	Semi-formalizable
	ST6		Self-contained	Formalizable
	ST7		Linked-explanatory	Formalizable
	ST8		Self-contained	Formalizable
	ST9		Self-contained	Formalizable
	ST10		Self-contained	Formalizable

Textual Expressions of Clause 56		Statement Type
Madde 56–Kapıcı Dairesi		*Id&heading*
A- Kapıcı Dairesi Yapılacak Binalar:		*subheading*
1	a) Konut, işyeri ve büro olarak kullanılacak binalardan (resmi daireler hariç);	Applicability Con.
2	1- Kaloriferli olanlarında, konut binalarının 12 daireden, fazla olanlarında kapıcı dairesi; işyeri ve büro binalarının 4000 m3 (brüt) ten fazla olanlarında bekçi odası ayrılması zorunludur.	Rule
3	2- Kalorifersiz olanlarda, konut binalarının 16 daireden fazla olmalarında kapıcı dairesi, işyeri ve büro binalarının 5000 m3 (brüt) ten fazla olanlarında bekçi odası ayrılması zorunludur.	Rule
4	b) Kaloriferli konut binalarının daire sayısı (50) den işyeri ve büro binalarının 15000 m3 (brüt) ten fazla olanlarında da kapıcı dairesinden veya bekçi odasından başka, kapıcı dairesi ölçü ve niteliğinde bir de kaloriferci dairesi ayrılacaktır.	Rule
5	c) Toplam daire sayısı 48'den fazla olan konut parsellerinde yukarıdaki şartlarda ikinci bir kapıcı dairesi aranır.	Rule
B- Kapıcı Dairelerinin ve Bekçi Odasının Ölçü ve Nitelikleri:		*subheading*
6	a) Kapıcı dairelerinin ve bekçi odalarının taban döşemesi üst kotu, tabii zeminden veya tesviye edilmiş bahçe kotundan aşağıda olamaz.	Rule
7	b) Kapıcı daireleri doğrudan ışık ve hava alabilecek şekilde düzenlenecek ve brüt alanı 45m2'den az olmayacaktır.	Rule
8	Minimum 12.00 m2 ve 7.00 m2'lik birer oda (odalardan biri ışıklıktan hava alabilir.) 4.00 m2'lik mutfak veya mutfak nişi ile 3.00 m2'lik WC + duş (ikisi aynı bölümde olabilir) ihtiva edecektir.	Rule
9	c) Bekçi odası en az 7.00 m2 büyüklüğünde ve doğrudan ışık ve hava alabilecek şekilde düzenlenecek ve en az 2.00 m2'lik bir WC + lavabo olacaktır.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C56	ST2		Self-contained	Formalizable
	ST3		Linked-explanatory	Formalizable
	ST4		Self-contained	Formalizable
	ST5		Linked-explanatory	Formalizable
	ST6		Self-contained	Formalizable
	ST7		Self-contained	Formalizable
	ST8		Self-contained	Formalizable
	ST9		Self-contained	Formalizable

Textual Expressions of Clause 57		Statement Type
Madde 57–Müştemilatlar		*Id&heading*
1	Binaların müştemilat kısımları *mümkün ise* binanın bodrum katında düzenlenir.	Clarification
2	Bahçede sığınak dışında müştemilat tertip edilemez.	Rule
3	Ayrıca , parsel içindeki yeri İmar Müdürlüklerince belirlenmek kaydıyla, belediyelerin ilgili teknik birimlerince hazırlanacak tip projeye göre her parsel için çöp toplama ünitesi yapılması zorunludur.	Out of Scope
4	Sobalı ısıtma sistemi seçilen yapılarda, her daire başına net 2.50 m2 lik kömürlük veya odunluk gibi müştemilatın tesisi zorunludur.	Rule
5	Bu hacim binanın bodrum katında veya müştemilat bölümünde de yapılabilir.	Clarification
6	Ancak, daire içinde veya kat sahanlığında yapılması halinde, max. (4.50) m2'yi geçemez ve bu alanın hava bacası veya ışıklıktan hava alması sağlanacaktır.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C57	ST2		Self-contained	Formalizable
	ST4		Self-contained	Formalizable
	ST6		Self-contained	Formalizable

Textual Expressions of Clause 59		Statement Type
Madde 59–Duvarlar	*Id&heading*	
1	Tüm binalarda; dış cepheye bakan duvar kalınlıkları (0.20)m.den küçük yapılamaz.	Rule
2	Ancak, “Isı Yalıtım Yönetmeliği” hükümleri saklı kalmak koşulu ile , hesapları (çözümleri) yapılarak yeni malzemeler kullanıldığında bu boyutlar değişebilir.	Rule
3	Bitişik nizama tabi yapıların bitişik duvarlarında ise, duvar kalınlığı (0.15)m. olabilir.	Rule
4	Teras çatılarda veya gizli çatılarda yapılacak parapet duvarlarının yüksekliği 1.00m.den az ve 1.10 m. den fazla olamaz.	Rule
5	Çatılarda kullanılan parapet duvarları yığma olarak yapıldığı taktirde, üzerlerine donatılı min.(0.30)m. yüksekliğinde hatıl dökülecek ve bu hatıl yer yer döşemeye bağlanacaktır.	Rule
6	Ayrıca kalkan duvarlarda, 2.00m.de yatay hatıl, 4.00m.de düşey hatıl (takviye kolon) yapılması zorunludur.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C59	ST1	R59.1	Self-contained	Formalizable
	ST2	R59.2	Linked-explanatory	Non-formalizable
	ST3	R59.3	Linked-explanatory	Formalizable
	ST4	R59.4	Self-contained	Formalizable
	ST5	R59.5	Self-contained	Formalizable
	ST6	R59.6	Self-contained	Formalizable

Textual Expressions of Clause 60		Statement Type
Madde 60–Bahçe Duvarları	*Id&heading*	
1	Bahçe duvarlarının yüksekliği, binaların yol tarafındaki cephe hatlarının önünde (0.50) m.yi gerisinde ise (1.00) m.yi geçemez.	Rule
2	Ayrıca üzerlerine yükseklikleri (1.00) m.yi aşmayan parmaklık yapılabilir.	Clarification
3	Fazla meyilli yerlerde uygulanacak şekli takdire belediye yetkilidir.	Rule
4	Okul, hastane, ceza evi, ibadet evi, elçilik, sefarethane, açık hava sineması ve benzerleri gibi özellik arz eden bina ve tesislerin bahçe duvarları ile sanayi bölgelerinde yapılacak bahçe ve çevre duvarları bu madde hükmüne tabi değildir.	Rule
5	Zemin katlarda dükkan yapılmasına izin verilen yapılarda, yaya kaldırımı ile aynı seviyedeki ön bahçeler *yayaya açık bulundurulacaktır*.	Rule
6	Bu bahçelerde yayaların *can emniyetini tehlikeye düşürecek* duvar ve manialar yapılamaz.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C60	ST1		Self-contained	Formalizable
	ST3		Linked-explanatory	Non-formalizable
	ST4		Linked-explanatory	Formalizable
	ST5		Linked-explanatory	Semi-formalizable
	ST6		Linked-explanatory	Semi-formalizable

Textual Expressions of Clause 65	Statement Type
Madde 65–Sığınak	*Id&heading*
Bayındırlık ve İskan Bakanlığınca çıkarılan “3194 Sayılı İmar Kanunu’na Göre Düzenlenmiş Sığınaklarla İlgili Yönetmelik”e uyulacaktır.	Rule

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C65	ST1		Self-contained	Fromalizable

Clause Id	Statement Id	Rule Id	Rule Type	
			Self-containedness	Formalizability
C66	ST2		Self-contained	Formalizable**
	ST3		Self-contained	Formalizable**
	ST5		Self-contained	Formalizable
	ST6		Self-contained	Formalizable
	ST7		Self-contained	Formalizable
	ST8		Self-contained	Formalizable
	ST9		Self-contained	Formalizable
	ST10		Self-contained	Formalizable
	ST11		Self-contained	Formalizable
	ST12		Self-contained	Formalizable**
	ST13		Self-contained	Formalizable
	ST14		Self-contained	Formalizable
	ST17		Self-contained	Formalizable
	ST18		Self-contained	Formalizable
	ST19		Self-contained	Formalizable
	ST20		Self-contained	Formalizable
	ST21		Self-contained	Formalizable
	ST22		Self-contained	Formalizable
	ST23		Self-contained	Formalizable
	ST24		Self-contained	Formalizable

	Textual Expressions of Clause 66	Statement Type
	Madde 66–Engelliler ile İlgili Hükümler *Id&heading*	
1	A- Katlı ve açık otoparklar, okullar, resmi binalar, havaalanı, gar, otogar, hastaneler, üniversiteler, sinema, tiyatro, kültürel yapılar, büyük marketler (hipermarketler) açık ve kapalı yüzme havuzları, stadyum termal tesisleri, kapalı spor salonları, açık spor salonları, yatak kapasitesi 100'den fazla olan oteller moteller ve yurt binaları, toplam inşaat alanı 1000 m2'den büyük sağlık tesisleri' nin projelendirme ve yapımı aşamasında aşağıdaki şartlara uyulacaktır.	Applicability Con.
2	a) Açık ve katlı otopark yeri olarak ayrılan alanın en az %2'si (1'den az olmamak kaydıyla*) engellilerin de kullanımını sağlayacak şekilde* düzenlenecektir.	Rule
3	Ayrıca bu oto yerleri 5 m. x 3.5 m. boyutunda olmalı, bina girişine *yakın* düzenlenmeli ve sarı renkle işaretlenmelidir.	Rule
4	Ayrıca, engellilere ait olduğu trafik park levhası ile belirlenmelidir.	Out of Scope
5	b) Bina girişlerinde; bedensel engellilerin kullanımı için, katlar arası düşey sirkülasyon sağlayan asansör veya merdivene ulaşan ve eğimi %6'yı aşmayan rampa yapılacaktır.	Rule
6	Rampanın iki yanına 90 cm yüksekliğinde ve 4-5 cm çapında yuvarlak tutunma barları yapılacaktır.	Rule
7	Rampa genişliği en az 90 cm olacak ve boşluk tarafına bordür konulacaktır.	Rule
8	c) Katlar arasında düşey sirkülasyonu sağlayan ana giriş merdivenlerinde; basamaklar çıkıntılı yapılmayacak, genişliği 30 cm'den az ve rıht yüksekliği ise 15 cm'den fazla yapılmayacaktır.	Rule
9	Duvar tarafına 90 cm yüksekliğinde tutunma barı yapılacaktır.	Rule
10	Tutunma barları ve merdiven küpeşteleri, ilk basamağa en az 30 cm dışından başlanmalıdır.	Rule
11	Döner merdiven yapılmalı ve en çok 10 basamakta bir dinlenme sahanlıkları tertiplenmelidir.	Rule
12	d) Bu binalardan asansör yapılması zorunlu olanlarında yapılacak asansörlerden en az bir tanesi bedensel engellilerin de kullanımını sağlayacak şekilde düzenlenecektir.	Rule
13	Bu asansörün önündeki sahanlık genişliği 150 cm'den az olamaz.	Rule
14	Asansör kabininin genişliği 110 cm'den, derinliği 140 cm'den, kapı genişliği 80 cm'den az olamaz.	Rule
15	Asansör düğmeleri 90-130 cm yükseklikte olmalı ve kabin içinde 90 cm yükseklikte tutunma barı düzenlenmelidir.	Out of Scope
16	Ayrıca bu asansörlerde görme engelliler için sesli ikaz sistemi tesis edilmeli ve kontrol düğmeleri kabartmalı yapılmalıdır.	Out of Scope
17	B- Okullar, üniversiteler, havaalanı, gar, otogar, stadyum, sinema, tiyatro toplam inşaat alanı 500 m2'yi aşan kültür yapıları, toplam inşaat alanı 1000 m2'yi aşan resmi yapılar, toplam inşaat alanı 1000 m2'yi aşan sağlık tesisleri ve dispanserlerde yapılacak WC'lerden en az birer adedi (bir kadın, bir erkek olmak üzere), bedensel engellilerin de kullanılabileceği şekilde düzenlenecektir.	Rule
18	Açık ve kapalı yüzme havuzları, açık spor alanları, kapalı spor alanlarında yapılacak WC'lerden ve duş mahallerinden en az birer adedi (bir kadın, bir erkek olmak üzere), bedensel engellilerin de kullanılabileceği şekilde düzenlenecektir.	Rule
19	Yatak kapasitesi 100'den fazla olan otel, motel, yurt binaları, hastanelerde; müşterek kullanılan WC'lerden en az birer adedi (bir kadın, bir erkek olmak üzere) ve içinde WC ve duş mahalli bulunan en az bir oda, bedensel engellilerin de kullanılabileceği şekilde düzenlenecektir.	Rule
20	Düzenlenen bu mekanlarda kapılar dışa açılacak, tutunma ve destek barları ile elçekleri yapılacaktır.	Rule
21	İç ölçüler WC'lerde 140 cm x 140 cm'den, WC + duş mekanlarında ise 160 cm x 220 cm'den az olamaz.	Rule
22	Duş mahalli en az 91,5 cm x 152,5 cm x 152,5 cm transfer oturaklı olarak düzenlenecektir.	Rule

APPENDIX B

RULE AND RULE-SET REPRESENTATIONS OF IMHZCODE

This appendix shows the representation of Izmir Municipality Housing and Zoning Code (IMHZCode) rule statements and rule-sets. Rule statements have been represented as rule objects in the form of structured data based on the pre-defined constructs. 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 attributes. Followings are the definition of each attribute, and tables present how rule statements are represented by using the constructs. A rule can be found by using rule id, which represent clause and statement number in Izmir Municipality Housing and Zoning Code (IMHZCode) 2013 version.

Definition of Attributes

Concept:

Name of the subject to which the rule applies.

Property:

Name of the attribute of the *concept.*

Comparator:

Name of the comparison operator such as “≥”, “≤”, “=”, “equal”, “boolean”

Value:

The specific value that is found in the code, whether numeric, descriptive, or Boolean.

Unit:

The unit of measure for numeric values.

		REQUIREMENT					SELECTION				
St.Id	**Rule Id**	**Concept**	**Property**	**C.**	**Value**	**U.**	**Concept**	**Property**	**Comp.**	**Value**	**U.**
ST2	R27.1	Setback	frontDistance	≥	5	m					
							Block	constructionOrder	equal	semiDetached	
ST4	R27.2	Setback	frontDistance	=	{Block_referencedFrontSetbackDistance}	m	Block	hasExistingBuilding	boolean	true	
							Block	constructionOrder	equal	plannedUnit	
ST5	R27.3	Setback	frontDistance	=	{Block_referencedFrontSetbackDistance}	m	Block	hasExistingBuilding	equal	true	
							Block	constructionOrder	equal	attached	
ST6	R27.4	Setback	frontDistance	=	{Block_referencedFrontSetbackDistance}	m	Block	is50%Developed	equal	true	
ST7	R27.5	Setback	sideDistance	=	3	m					
ST8	R27.6	Setback	sideDistance	=	(3+((:{building_numberofStorey}:-4)/2))	m	Building	numberofStorey	≥	4	
ST9	R27.7	Setback	sideDistance	≥	5	m	Building	constructionTechnique	equal	timberFramed	
ST10	R27.8	Setback	rearDistance	=	(:{building_height}:/2)	m					
ST13	R27.9	Setback	rearDistance	≥	3	m	Block	hasExistingBuilding	boolean	true	
							Block	constructionOrder	equal	semiDetached	
ST14	R27.10	Setback	rearDistance	=	{Block_referencedRearSetbackDistance}	m	Block	hasExistingBuilding	boolean	true	
							Block	constructionOrder	equal	plannedUnit	
ST15	R27.11	Setback	rearDistance	=	{Block_referencedRearSetbackDistance}	m	Block	hasExistingBuilding	boolean	true	
							Block	constructionOrder	equal	attached	
ST16	R27.12	Setback	rearDistance	=	{Block_referencedRearSetbackDistance}	m	Block	is50%Developed	boolean	true	

Rule-set 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)))

		REQUIREMENT					SELECTION				
St.Id	**Rule Id**	**Concept**	**Property**	**C.**	**Value**	**U.**	**Concept**	**Property**	**Comp.**	**Value**	**U.**
ST1	R28.1	Building	depth	≤	22	m					
ST2	R28.2	Building	depth	≤	22	m	Block	hasExistingBuilding	equal	true	
ST3	R28.3	Building	depth	=	{Block_refBuildingdepth}		Block	constructionOrder	equal	semiDetached	
ST4	R28.4	Building	depth	=	{Block_refBuildingdepth}		Block	constructionOrder	equal	plannedUnit	
ST5	R28.5	Building	depth	=	{Block_refBuildingdepth}		Block	constructionOrder	equal	attached	
ST6	R28.6	Building	depth	=	{Block_refBuildingdepth}		Lot	onCorner	boolean	true	
ST8	R28.8	Building	depth	≤	{Lot_clearDepth}		Lot	numberofFacingRoad	=	2	

Rule-set Id	**Subject**		**Set**
	Concept	**Property**	
RS28.A	Building	depth	(‖: R28.1, (&: R28.2, (‖: R28.3, R28.4, R28.5)), R28.6, R28.8)

		REQUIREMENT					SELECTION				
St.Id	**Rule Id**	**Concept**	**Property**	**C.**	**Value**	**U.**	**Concept**	**Property**	**Comp.**	**Value**	**U.**
ST1	R29.1	Building	facade	≤	30	m	Block	constructionOrder	equal	detached	

Rule-set Id	**Subject**		**Set**
	Concept	**Property**	
RS29.A	Building	facade	R29.1

		REQUIREMENT					SELECTION				
St.Id	**Id**	**Concept**	**Property**	**C.**	**Value**	**U.**	**Concept**	**Property**	**Comp.**	**Value**	**U.**
ST2	R30.1	Building	height	≤	3.80	m	1StroeyBuilding				
ST3	R30.2	Building	height	≤	6.80	m	2StroeyBuilding				
ST4	R30.3	Building	height	≤	9.80	m	3StroeyBuilding				
ST5	R30.4	Building	height	≤	12.80	m	4StroeyBuilding				
ST6	R30.5	Building	height	≤	15.80	m	5StroeyBuilding				
ST7	R30.6	Building	height	≤	18.80	m	6StroeyBuilding				
ST8	R30.7	Building	height	≤	21.80	m	7StroeyBuilding				
ST9	R30.8	Building	height	≤	24.80	m	8StroeyBuilding				
ST10	R30.9	Building	height	≤	77.80	m	9StroeyBuilding				
ST11	R30.10	Building	height	≤	30.80	m	10StroeyBuilding				

Id	**Subject**		**Set**
	Concept	**Property**	
RS30.A	Building	height	(||: R30.1, R30.2, R30.3, R30.4, R30.5, R30.6, R30.7, R30.8, R30.9, R30.10)

		REQUIREMENT					SELECTION				
St.Id	**Id**	**Concept**	**Property**	**C.**	**Value**	**U.**	**Concept**	**Property**	**Comp.**	**Value**	**U.**
ST1	R38.1.1	Storey	distanceToProjectZero	≥	0.50	m	Basement				
ST1	R38.1.2	Storey	distanceToProjectZero	≤	1.00	m					
							Basement				
ST2	R38.2	Storey	distanceToProjectZero	≥	0.00	m	Zone	usage	equal	store || parking	

Id	**Subject**		**Set**
	Concept	**Property**	
RS38.A	Storey	distanceToProjectZero	(&: (||: R38.1.1, R38.2), R38.1.2)

		REQUIREMENT					SELECTION				
St.Id	Id	Concept	Property	C.	Value	U.	Concept	Property	Comp.	Value	U.
ST1	R39.1	Building	constructionTechnique	!equal	timberFramed		Block	constructionOrder	equal	detached	
ST2	R39.2	Building	constructionTechnique	!equal	timberFramed		Setback	sideDistance	<	5	m
ST3	R39.3	Building	constructionTechnique	!equal	timberFramed		Building	height	>	6.80	m
							Block	constructionOrder	equal	attached	
ST4	R39.4	Building	constructionTechnique	!equal	timberFramed		Building	hasFireWall	boolean	false	

Id	Subject		Set
	Concept	Property	
RS39.A	Building	constructionTechnique	(&: (||: R39.1, R39.4), R39.2, R39.3)

		REQUIREMENT					SELECTION				
St.Id	Id	Concept	Property	C.	Value	U.	Concept	Property	Comp.	Value	U.
ST1	R40.1.1	Eave	homeStorey	equal	last						
ST1	R40.1.2	Eave	width	≤	0.50	m					
ST2	R40.2.1	Canopy	width	≤	1.50	m	Lot	isWithFrontGarden	boolean	false	
ST2	R40.2.2	Canopy	isConsole	Boolean	True		Lot	isWithFrontGarden	boolean	false	
ST2	R40.2.3	Canopy	distancetoLot	≤	2.00	m	Lot	isWithFrontGarden	boolean	false	
ST3	R40.3.1	Canopy	level	≥	3.00	m	Lot	isWithFrontGarden	boolean	false	
ST3	R40.3.2	Canopy	width	≤	{Block_sidewalkWidth}	m	Lot	isWithFrontGarden	boolean	false	
ST4	R40.4.1	Canopy	width	≤	{Setback_frontDistance}	m	Lot	isWithFrontGarden	boolean	true	
ST4	R40.4.2	Canopy	isConsole	boolean	true		Lot	isWithFrontGarden	boolean	true	

Id	Subject		Set
	Concept	Property	
RS40.A	Eave	homeStorey	R40.1.1
RS40.B	Eave	width	R40.1.2
RS40.C	Canopy	width	(||: (&: R40.2.1, 40.3.2), R40.4.1)
RS40.D	Canopy	isConsole	(||: R40.2.2, R40.4.2)
RS40.E	Canopy	distancetoLot	R40.2.3
RS40.F	Canopy	level	R40.3.1

		REQUIREMENT					SELECTION				
St.Id	Id	Concept	Property	C.	Value	U.	Concept	Property	Comp.	Value	U.
ST2	R41.2	Roof	pitch	=	33	%					
							Block	constructionOrder	equal	detached	
							Building	type	equal	dublexHouse	
ST3	R41.3.1	Roof	pitch	=	{Roof_pitch}	%	Roof	run	≤	{Roof_calculatedRun}	
							Block	constructionOrder	equal	detached	
							Building	type	equal	dublexHouse	
ST3	R41.3.2	Roof	form	equal	{Roof_form}		Roof	run	≤	{Roof_calculatedRun}	m
							Building	type	!equal	dublexHouse	
ST7	R41.4	Roof	form	equal	hip		Roof	run	≤	3	m
ST8	R41.5	Zone	hasSpace	equal	false		attic				
							attic				
ST9	R41.6.1	Zone	hasSpace	boolean	true		Zone	{containingSpace_usage}	equal	liftShaft || flueShaft	
							attic				
							Building	type	equal	dublexHouse	
							Building	height	≤	6.80	m
							Zone	{containingSpace_hasConnectedtoIUnit	boolean	true	
							Zone	{containingSpace _boundary}	≤	{containingSpace_zone_boundary}	
ST9	R41.6.2	Zone	hasSpace	boolean	true		Zone	{containingSpace _area}	≤	(:{space_zone _area}:*30/100)	m2
ST10	R41.7.1	Zone	hasSpace	boolean	true		Zone	{containingSpace _isOccupied}	boolean	false	
ST10	R41.7.2	Zone	hasSpace	boolean	true		Zone	{containingSpace _height}	≥	1.80	m
ST11	R41.8	Zone	hasSpace	boolean	true		Zone	{containingSpace _relatedSpace}	!equal	terrace	
							Ceiling	homeStorey	!equal	lastFloor	
ST13	R41.10	Ceiling	isSloped	boolean	false		Ceiling	level	≥	{Roof_run}	
ST14	R41.11	Wall	exceedingLmtfromRoof	≤	0.60	m	Wall	type	equal	gable	
ST15	R41.12.1	Roof	hasExtension	boolean	false						
ST15	R41.12.2	Roof	hasExtension	boolean	true		Roof	{exceedingSpace_usage}	equal	flueShaft	
ST15	R41.12.3	Roof	hasExtension	boolean	true		Roof	{exceedingSpace_usage}	equal	airShaft	
							Roof	{exceedingSpace_height}	≤	2.10	m
ST16	R41.13	Roof	hasExtension	boolean	true		Roof	{exceedingSpace_usage}	equal	stairShaft	
							terrace				
ST17	R41.14.1	Zone	hasSpace	boolean	true		Zone	{containingSpace_usage}	equal	liftShaft	
							terrace				
							Zone	{containingSpace _usage}	equal	stairShaft	
ST17	R41.14.2	Zone	hasSpace	boolean	true		Zone	{containingSpace _height}	≤	2.10	

		REQUIREMENT					SELECTION				
St.Id	**Id**	**Concept**	**Property**	**C.**	**Value**	**U.**	**Concept**	**Property**	**Comp.**	**Value**	**U.**
ST1	R42.1.1	Building	hasCantilever	boolean	false						
							Cantilever	distancetoLotFrontBorder	≥	0.00	m
							Cantilever	width	≤	1.50	m
							Cantilever	distancetoProjectZero	≥	2.40	m
ST1	R42.1.2	Building	hasCantilever	boolean	true		Cantilever	distancetoLotSideBorder	≥	3.00	m
							Cantilever	distancetoLotRearBorder	≥	3.00	m
ST2	R42.2	Building	hasCantilever	boolean	true		Cantilever	length	≤	{building_facade}	
							Lot	{Setback_sideDistance}	≥	3.00	m
							Lot	{Setback_sideDistance}	≤	4.00	m
							Cantilever	distancetoLotSideBorder	≥	2.00	m
							Cantilever	width	≤	1.00	m
							Cantilever	type	equal	open	
ST3	R42.3	Building	hasCantilever	boolean	true		Cantilever	length	≤	(:{building_facade}:/3)	m
							Block	constructionOrder	equal	attached	
ST4	R42.4	Building	hasCantilever	boolean	true		Cantilever	distancetoLotSideBorder	≥	2.00	m
							Block	constructionOrder	equal	attached	
							existingBuilding	cantilerDistancetoBorder	=	0.00	m
ST6	R42.6	Building	hasCantilever	boolean	true		Cantilever	distancetoLotSideBorder	=	0.00	m
							Lot	roadWidth	≥	6.00	m
							Lot	isWithFrontGarden	boolean	false	
ST7	R42.7	Building	hasCantilever	boolean	true		Cantilever	width	=	((:{Lot_roadWidth}:-6)/2)	m
							Cantilever	width	≤	1.00	m
ST8	R42.8	Building	hasCantilever	boolean	true		Cantilever	distancetoProjectZero	≥	3.00	m
ST10	R42.10	Building	hasCantilever	boolean	true		Cantilever	width	≤	0.20	m
							Block	facedtoGreenArea	boolean	true	
ST11	R42.11	Building	hasCantilever	boolean	true		Cantilever	width	≤	1.00	m

Id	**Subject**		**Set**
	Concept	**Property**	
RS42.A	Building	hasCantilever	(||: R42.1.1, (&: R42.1.1, (||: R42.2, R42.3, R42.6)), (&: (||: R42.7, R42.11), R42.8), R42.10)

		REQUIREMENT					SELECTION				
St.Id	**Id**	**Concept**	**Property**	**C.**	**Value**	**U.**	**Concept**	**Property**	**Comp.**	**Value**	**U.**
							Building	usage	equal	dwelling	
ST1	R43.1	Space	hasOpeningTo	equal	outside		Space	usage	equal	{livingRoom || bedroom}	
							Building	usage	equal	dwelling	
ST2	R43.2.1	Space	hasOpeningTo	equal	outside		Space	usage	equal	{kitchen || room || bathroom || wc}	
							Building	usage	equal	dwelling	
ST2	R43.2.2	Space	hasOpeningTo	equal	lightShaft		Space	usage	equal	{kitchen || room }	
							Building	usage	equal	dwelling	
ST2	R43.2.3	Space	hasOpeningTo	equal	ventilationShaft		Space	usage	equal	{bathroom || wc }	
							Building	usage	equal	dwelling	
ST3	R43.3.1	lightShaft	width	≥	1.00	m	Building	numberofStorey	≤	2	
							Building	usage	equal	dwelling	
ST3	R43.3.2	lightShaft	area	≥	3.00	m2	Building	numberofStorey	≤	2	
							Building	usage	equal	dwelling	
ST4	R43.4.1	lightShaft	width	≥	1.50	m	Building	numberofStorey	≤	5	
							Building	usage	equal	dwelling	
ST4	R43.4.2	lightShaft	area	≥	4.50	m2	Building	numberofStorey	≤	5	
							Building	usage	equal	dwelling	
ST5	R43.5.1	lightShaft	width	≥	2.00	m	Building	numberofStorey	≤	9	
							Building	usage	equal	dwelling	
ST5	R43.5.2	lightShaft	area	≥	6.00	m2	Building	numberofStorey	≤	9	
							Building	usage	equal	dwelling	
ST6	R43.6.1	lightShaft	width	≥	2.00	m	Building	numberofStorey	≥	10	
							Building	usage	equal	dwelling	
ST6	R43.6.2	lightShaft	area	≥	9.00	m2	Building	numberofStorey	≥	10	
ST8	R43.8.1	lightShaft	width	≥	1.50	m	Building	numberofStorey	≤	2	
ST8	R43.8.2	lightShaft	area	≥	4.50	m2	Building	numberofStorey	≤	2	
ST9	R43.9.1	lightShaft	width	≥	2.00	m	Building	numberofStorey	≤	9	
ST9	R43.9.2	lightShaft	area	≥	6.00	m2	Building	numberofStorey	≤	9	
ST10	R43.10.1	lightShaft	width	≥	2.00	m	Building	numberofStorey	≥	10	
ST10	R43.10.2	lightShaft	area	≥	9.00	m2	Building	numberofStorey	≥	10	
ST11	R43.11.1	ventilationShaft	width	≥	0.60	m					
ST11	R43.11.2	ventilationShaft	area	≥	0.36	m2					
ST11	R43.11.3	ventilationShaft	width	≥	0.80	m	ventilationShaft	hasInstallation	boolean	true	
ST11	R43.11.4	ventilationShaft	area	≥	0.64	m2	ventilationShaft	hasInstallation	boolean	true	

Id	Subject		Set
	Concept	Property	
RS43.A	Space	hasOpeningTo	(‖: R43.1, R43.2.1, R43.2.2, R43.2.2)
RS43.B	lightShaft	width	(‖: R43.3.1, R43.4.1, R43.5.1, R43.6.1, R43.8.1, R43.9.1, R43.10.1)
RS43.C	lightShaft	area	(‖: R43.3.2, R43.4.2, R43.5.2, R43.6.2, R43.8.2, R43.9.2, R43.10.2)
RS43.D	ventilationShaft	width	(‖: R43.11.1, R43.11.3)
RS43.E	ventailationShaft	area	(‖: R43.11.2, R43.11.4)

		REQUIREMENT					SELECTION				
St.Id	Id	Concept	Property	C.	Value	U.	Concept	Property	Comp.	Value	U.
ST1	R44.1.1	Zone	containingSpaceList	include	livingRoom		dwelling				
ST1	R44.1.2	Zone	containingSpaceList	include	bedroom		dwelling				
ST1	R44.1.3	Zone	containingSpaceList	include	bedNische		dwelling				
ST1	R44.1.4	Zone	containingSpaceList	include	kitchen		dwelling				
ST1	R44.1.5	Zone	containingSpaceList	include	cookNische		dwelling				
ST1	R44.1.6	Zone	containingSpaceList	include	bathroom		dwelling				
ST1	R44.1.7	Zone	containingSpaceList	include	bathNische		dwelling				
ST1	R44.1.8	Zone	containingSpaceList	equal	wc		dwelling				
ST3	R44.2.1	Space	width	≥	3.00	m	livingRoom				
ST4	R44.2.2	Space	area	≥	12.00	m2	livingRoom				
ST4	R44.3.1	Space	width	≥	2.60	m	bedroom				
ST4	R44.3.2	Space	area	≥	7.28	m2	bedroom				
ST5	R44.4.1	Space	width	≥	1.50	m	bedNiche				
ST5	R44.4.2	Space	area	≥	3.00	m2	bedNiche				
ST6	R44.5.1	Space	width	≥	1.50	m	kitchen				
ST6	R44.5.2	Space	area	≥	3.60	m2	kitchen				
ST7	R44.6.1	Space	width	≥	0.70	m	cookNische				
ST7	R44.6.2	Space	area	≥	1.40	m2	cookNische				
ST8	R44.7.1	Space	width	≥	1.20	m	bathroom				
ST8	R44.7.2	Space	area	≥	3.48	m2	bathroom				
ST9	R44.8.1	Space	width	≥	1.20	m	bathNische				
ST9	R44.8.2	Space	area	≥	2.64	m2	bathNische				
ST10	R44.9.1	Space	width	≥	0.90	m	wc				
ST10	R44.9.2	Space	area	≥	1.08	m2	wc				

ST11	R44.10.1	Space	width	≥	1.00	m	entrance				
ST11	R44.10.2	Space	area	≥	1.32	m2	entrance				
ST12	R44.11.1	Space	width	≥	1.10	m	corridor				
ST12	R44.11.2	Space	area	≥	1.32	m2	corridor				
ST13	R44.12.1	Space	width	≥	2.10	m	room				
ST13	R44.12.2	Space	area	≥	6.00	m2	room				
ST14	R44.13	Space	containedSpace	equal	livingRoom		bathNische				
ST15	R44.14.1	Space	connectedTo	equal	chimney		cookNische				
ST15	R44.14.2	Space	connectedTo	equal	ventilationShaft		cookNische				
ST16	R44.15.1	Zone	containingSpaceList	equal	coalCellar		building	typeofHeatingSystem	equal	stove	
ST16	R44.15.2	Space	area	≥	2.50	m2	coalCellar				
							coalCellar				
ST17	R44.16	Space	area	≥	4.50	m2	Space	containedZone	equal	iUnit	
ST18	R44.17	Space	isFurnished	boolean	true		wetSpace				
							building	type	equal	public	
							corridor				
ST19	R44.18.1	Space	width	≥	2.00	m	Space	length	≤	20.00	m
							building	type	equal	public	
							corridor				
ST19	R44.18.2	Space	width	≥	2.50	m	Space	length	≥	20.00	m
ST20	R44.19	Zone	facade	≥	2.00	m	workingPlace				

Id	**Subject**		**Set**
	Concept	**Property**	
RS44.A	Zone	containingSpaceList	(&: R44.1.1, (||: R44.1.2, R44.1.3), (||: R44.1.4, R44.1.5), (||: R44.1.6, (&: R44.1.7, R44.1.8)), R44.15.1)
RS44.B	Space	width	(||: R44.2.1, R44.3.1, R44.4.1, R44.5.1, R44.6.1, R44.7.1, R44.8.1, R44.9.1, R44.10.1, R44.11.1, R44.12.1, R44.18.1, R44.18.2)
RS44.C	Space	area	(||: R44.2.2, R44.3.2, R44.4.2, R44.5.2, R44.6.2, R44.7.2, R44.8.2, R44.9.2, R44.10.2, R44.11.2, R44.12.2, R44.15.2, R44.16)
RS44.D	Space	containedSpace	R44.13
RS44.E	Space	connectedTo	(&: R44.14.1, R44.14.2)
RS44.F	Space	isFurnished	R44.17
RS44.G	Zone	facade	R44.19

		REQUIREMENT					SELECTION				
St.Id	**Id**	**Concept**	**Property**	**C.**	**Value**	**U.**	**Concept**	**Property**	**Comp.**	**Value**	**U.**
ST1	R45.1	Storey	height	≥	2.60	m					
ST3	R45.2.1	Space	height	≥	2.20	m	Space	usage	equal	{bathNische || bathroom || wc || cellar || office || entrance || corridor || innerCorridor || }	
ST3	R45.2.2	Storey	height	≥	2.20	m	Storey	isOccupied	boolean	false	
							Storey	level	equal	underground	
ST3	R45.2.3	Storey	height	≥	2.20	m	Building	usage	equal	auxiliary	
ST4	R45.3	Space	height	≥	2.00	m	Space	usage	equal	garage	
ST5	R45.4	Storey	height	≥	2.60	m	building	isOccupied	boolean	true	
ST6	R45.5	Space	height	≥	3.00	m	Space	{containedZone_usage}	equal	{store || cafe || restaurant}	
ST7	R45.6	Space	height	≥	3.50	m	Space	{containedZone_usage}	equal	{wedding hall || casino || disco || pub}	
ST8	R45.7	Space	height	≥	2.60	m	Space	{containedZone_usage}	equal	{wedding hall || casino || disco || pub}	
							Space	usage	equal	serviceArea	
ST9	R45.8	Space	area	≤	((:{containedZone_area)*0.5))	m	Space	{containedZone_usage}	equal	{wedding hall || casino || disco || pub}	
							Space	usage	equal	serviceArea	

Id	**Subject**		**Set**
	Concept	**Property**	
RS45.A	Storey	height	(||: R45.1, R45.2.2, R45.2.3, R45.4)
RS45.B	Space	height	(||: R45.2.1, R45.3, R45.5, R45.6, R45.7)
RS45.C	Space	area	R45.8

		REQUIREMENT					SELECTION				
St.Id	Id	Concept	Property	C.	Value	U.	Concept	Property	Comp.	Value	U.
ST1	R46.1.1	Window	width	≥	0.60	m					
ST1	R46.1.2	Window	area	≥	(:{relatedSpace_area}:/8)	m	Window	{relatedSpace_usage}	equal	{livingRoom || room || kitchen}	
ST1	R46.1.3	Window	area	≥	1.25	m2					
						m2	building	type	equal	dublexHouse	
ST3	R46.2.1	Window	area	≤	0.80		Window	{relatedSpace_containedZone}	equal	attic	
							building	type	equal	dublexHouse	
ST3	R46.2.2	Space	numberOfWindow	=	2		Space	containedZone	equal	attic	

Id	Subject		Set
	Concept	Property	
RS46.A	Window	width	R46.1.1
RS46.B	Window	area	(&: R46.1.2, R46.1.3)
RS46.C	Space	numberofWindow	R46.2.2

		REQUIREMENT					SELECTION				
St.Id	Id	Concept	Property	C.	Value	U.	Concept	Property	Comp.	Value	U.
ST1	R47.1	Door	height	≥	2.10	m					
ST2	R47.2	Door	width	≥	1.30	m	mainEntranceDoor				
ST3	R47.3	Door	width	≥	1.00	m	entranceDoor				
ST4	R47.4.1	Door	width	≥	0.90	m	roomDoor				
ST4	R47.4.2	Door	width	≥	0.90	m	kitchenDoor				
ST5	R47.5.1	Door	width	≥	0.80	m	bathroomDoor				
ST5	R47.5.2	Door	width	≥	0.80	m	wcDoor				
ST5	R47.5.3	Door	width	≥	0.80	m	cellarDoor				
ST6	R47.6	Door	width	≥	1.00	m	storeDoor				
ST8	R47.8	Door	allowAirTransfer	boolean	true		bathroomDoor				

Id	Subject		Set
	Concept	Property	
RS47.A	Door	height	R47.1
RS47.B	Door	width	(‖: R47.2, R47.3, R47.4.1, R47.4.2, R47.5.1, R47.5.2, R47.5.3, R47.6)
RS47.C	Door	allowAirTransfer	R47.8

		REQUIREMENT					SELECTION				
St.Id	Id	Concept	Property	C.	Value	U.	Concept	Property	Comp.	Value	U.
							Building	height	≥	12.80	
							Building	numberofStorey	≥	4	
ST1	R48.1.1	Building	numberofLift	≥	1		Building	usage	equal	dwelling	
							Building	height	≥	12.80	
							Building	numberofStorey	≥	2	
ST1	R48.1.2	Building	numberofLift	≥	1		Building	usage	!equal	dwelling	
ST1	R48.1.3	Lift	startupStorey	equal	ground						
ST1	R48.1.4	Lift	finishingStorey	equal	last						
ST1	R48.1.5	Lift	hasAccessToEveryStorey	boolean	true						
ST1	R48.1.6	Lift	complyWith	equal	"TSE & AY"						
							Building	height	≥	12.80	
							Building	usage	equal	dwelling	
ST3	R48.2	Building	numberofLift	≥	2		Building	numberofIUnit	≥	20	

Id	Subject		Set
	Concept	Property	
RS48.A	Building	numberofLift	(‖ R48.1.1, R48.1.2, R48.2)
RS48.B	Lift	startupStorey	R48.1.3
RS48.C	Lift	finishningStorey	R48.1.4
RS48.D	Lift	hasAccessToEveryStorey	R48.1.5
RS48.D	Lift	complyWith	R48.1.6

		REQUIREMENT					SELECTION				
St.Id	Id	Concept	Property	C.	Value	U.	Concept	Property	Comp.	Value	U.
ST1	R49.1.1	Building	numberofStair	≥	1						
ST1	R49.1.2	Stair	material	!equal	wood		Stair	usage	equal	main	
ST2	R49.2	Stair	holeWidth	≥	20	cm					
ST3	R49.3	Space	hasOpeningTo	equal	outside		Space	usage	equal	stairWell	
ST4	R49.4.1	Stair	hasAccesToBasement	boolean	true		Stair	usage	equal	main	
ST4	R49.4.2	Stair	hasAccesToRoof	boolean	true		Stair	usage	equal	main	
ST5	R49.5.1	Stair	flightWidth	≥	1.20	m					
ST5	R49.5.2	Stair	landingWidth	≥	1.20	m					
ST6	R49.6.1	Stair	riserHeight	=	((60-:{threadLength}:)/2)						
ST6	R49.6.2	Stair	threadLegth	=	(60-(2*:{riserHeight}:))						
ST9	R49.8	Stair	landingWidth	≥	flightWidth						
ST10	R49.9	Stair	landingWidth	≥	1.00	m	Stair	type	equal	{straight || spiral}	
ST12	R49.11.1	Stair	flightWidth	≥	1.20	m	Building	usage	equal	dwelling	
ST12	R49.11.2	Stair	landingWidth	≥	1.20	m	Building	usage	equal	dwelling	
ST13	R49.12.1	Stair					Building	usage	equal	dwelling	
			flightWidth	≥	0.90	m	Stair	homeStorey	equal	basement	
ST13	R49.12.2	Stair					Building	usage	equal	dwelling	
			flightWidth	≥	0.90	m	Stair	usage	equal	service	
ST13	R49.12.3	Stair					Building	usage	equal	dwelling	
			landingWidth	≥	0.90	m	Stair	homeStorey	equal	basement	
ST13	R49.12.4	Stair					Building	usage	equal	dwelling	
			landingWidth	≥	0.90	m	Stair	usage	equal	service	
ST15	R49.14.1	Stair	riserHeight	≤	0.175	m	Building	usage	equal	dwelling	
ST15	R49.14.2	Stair	threadLegth	≥	0.28	m	Building	usage	equal	dwelling	
ST15	R49.14.3	Stair	threadLegth Min	≥	0.10	m	Building	usage	equal	dwelling	
ST16	R49.15.1	Stair	flightWidth	≥	1.60	m	Building	usage	!equal	dwelling	
ST16	R49.15.2	Stair	landingWidth	≥	1.60	m	Building	usage	!equal	dwelling	
ST17	R49.16	Stair	hasRailBothSide	boolean	true		Building	usage	!equal	dwelling	
ST18	R49.17.1	Stair	riserHeight	≤	0.16	m	Building	usage	!equal	dwelling	
ST18	R49.17.2	Stair	threadLegth	≥	0.30	m	Building	usage	!equal	dwelling	
ST18	R49.17.3	Stair	threadLegth Min	≥	0.125	m	Building	usage	!equal	dwelling	
ST19	R49.18.1	Stair					Building	usage	!equal	dwelling	
			flightWidth	≥	1.20	m	Stair	usage	!equal	main	
ST19	R49.18.2	Stair					Building	usage	!equal	dwelling	
			landingWidth	≥	1.20	m	Stair	usage	!equal	main	
ST19	R49.18.3	Stair	riserHeight	≤	0.175	m	Building	usage	!equal	dwelling	

							Stair	usage	!equal	main
ST19	R49.18.4	Stair				m	Building	usage	!equal	dwelling
			threadLegth	≥	0.28		Stair	usage	!equal	main
ST19	R49.18.5	Stair				m	Building	usage	!equal	dwelling
			threadLegth Min	≥	0.125		Stair	usage	!equal	main

Id	**Subject**		**Set**
	Concept	**Property**	
RS49.A	Building	numberofStair	R49.1.1
RS49.B	Stair	material	R49.1.2
RS49.C	Stair	holeWidth	R49.2
RS49.D	Space	hasOpeningTo	R49.3
RS49.E	Stair	hasAccesToBasement	R49.4.1
RS49.F	Stair	hasAccesToRoof	R49.4.2
RS49.G	Stair	flightWidth	(||: R49.5.1, R49.11.1, R49.12.1, R49.12.2, R49.15.1, R49.18.1)
RS49.H	Stair	landingWidth	(||: R49.5.2, R49.8, R49.9, R49.11.1, R49.12.3, R49.12.4, R49.15.2, R49.18.2)
RS49.I	Stair	riserHeight	(&: R49.6.1, (||: R49.14.1, R49.17.1, R49.18.3))
RS49.J	Stair	threadLegth	(&: R49.6.2, (||: R49.14.2, R49.17.2, R49.18.4))
RS49.K	Stair	threadLegth Min	(||: R49.14.3, R49.17.3, R49.18.5)
RS49.L	Stair	hasRailBothSide	R49.16

		REQUIREMENT					SELECTION				
St.Id	**Id**	**Concept**	**Property**	**C.**	**Value**	**U.**	**Concept**	**Property**	**Comp.**	**Value**	**U.**
ST1	R50.1	fireEscape	complyWith	equal	“BinalarınYangındanKorunması HakkındaYönetmelik”						
ST2	R50.2.1	fireEscape	width	≥	90	cm	building	usage	equal	{dwelling || office}	
ST2	R50.2.2	fireEscape	width	≥	120	cm	building	type	equal	public	

Id	**Subject**		**Set**
	Concept	**Property**	
RS50.A	fireescape	complyWith	R50.1
RS50.B	fireescape	width	(||: R50.2.1, R50.2.2)

		REQUIREMENT					SELECTION				
St.Id	**Id**	**Concept**	**Property**	**C.**	**Value**	**U.**	**Concept**	**Property**	**Comp.**	**Value**	**U.**
ST1	R51.1.1	terrace	hasBalustrade	boolean	true		terrace	type		{Roof || Storey}	
ST1	R51.1.2	balcony	hasBalustrade	boolean	true						
ST1	R51.1.3	Stair	hasBalustrade	boolean	true		Stair	numberofThread	≥	5	
ST1	R51.1.4	blustrade	height	≥	1.00	m					
ST2	R51.2	blustrade	barDistance	≤	10	cm					
ST2	R51.3	blustrade	hasProtection	Boolean	true		blustrade	type	equal	horizontal	

Id	**Subject**		**Set**
	Concept	**Property**	
RS51.A	terrace	hasBalustrade	R51.1.1
RS51.B	balcony	hasBalustrade	R51.1.2
RS51.C	Stair	hasBalustrade	R51.1.3
RS51.D	blustrade	height	R51.1.4
RS51.E	blustrade	barDistance	R51.2
RS51.F	blustrade	hasProtection	R51.3

VITA

PERSONAL

Surname, Name	**:** MACİT Sibel
Date and Place of Birth	**:** 16.12.1975 – İzmir (Turkey)
E-mail	**:** sibelmacit@gmail.com

EDUCATION

PhD., İzmir Institute of Technology, Graduate School of Engineering and Sciences, Department of Architecture (2008-2014) – Supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK) via the 2214 -Abroad Research Grant Programme.

Thesis: "Computer Representation of Building Codes for Automated Compliance Checking"

M.Sc., Balıkesir University, Graduate School of Natural and Applied Sciences, Department of Architecture (2004-2007)

Thesis: "Interoperability between Computer Aided Architectural Design Environments and a Room Acoustics Analysis Application using ifcXML"

B.Arch., Balıkesir University, Faculty of Engineering and Architecture, Department of Architecture (1993-1998)

ACADEMIC EXPERIENCES

Research Assistant, Balıkesir University, Faculty of Engineering and Architecture, Department of Architecture (2001-2008)

Research Assistant, İzmir Institute of Technology, Graduate School of Engineering and Sciences, Department of Architecture (2008-2014)

Guest Researcher, Vienna University of Technology, Institute of Architectural Sciences, Department of Digital Architecture and Planning (2011-2012)