PARAMETRIC URBAN REGULATION MODELS

FOR PREDICTING DEVELOPMENT PERFORMANCES

A Dissertation

by

JONG BUM KIM

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Mark J. Clayton
Committee Members,	Geoffrey J. Booth
	Valerian Miranda
	Wei Yan
Head of Department,	Ward V. Wells

December 2014

Major Subject: Architecture

ABSTRACT

This research developed and evaluated the Parametric Urban Regulation Model (PURM) to represent urban regulations in parametric Building Information Modeling (BIM) and assess the development performances of urban regulations prior to the urban regulation adoption. The PURM was formed with the Parametric Urban Design Model (PUDM), the Parametric Urban Design Model Object (PUDO), and the Parametric Urban Design Application (PUDA).

The first contribution, representing urban regulation provisions in parametric BIM, was driven by parameterization of the urban regulation provisions with the PUDM and the PUDO. By using the parametric modeling within BIM technology, five types of PUDOs of Site, Block, Parcel, Building, and Parking were assembled to create the PUDM as a 3D urban regulation plan. The PUDOs and the PUDM visualized form implication of urban regulation provisions with the object geometry in parametric BIM.

The second contribution, testing development performances of the urban regulation provisions, was devised to articulate the advantage of the urban regulation modeling in parametric BIM. The geometrical attributes of PUDOs were expressed with the parametric relationships, so the PUDM could present a range of regulation provision values. Once the PUDM was built in parametric BIM, an energy performance analysis could be performed. The PUDAs enabled the economic analysis based on the simplified pro forma estimation method.

The third contribution, reducing ambiguity in interpretation of the urban regulation provisions, was experimented with an existing zoning regulation and 11 software prototypes of the PUDAs. Some associations among the provisions that can make the regulation interpretation complex were imbedded in the PUDAs so that a set of related provisions can be determined

simultaneously. The environmental and the economic analyses made the relations among the provision values and the development performances explicit.

In the long run, the PURM can achieve benefits in comparison to conventional methods of representing urban regulations. The development performances can be assessed in explicit and direct ways, which were often unforeseen and unintended in the current practice. The PURM can potentially contribute the new platform development that encapsulates the urban plan without a static regulation tool and that improves the quality of urban planning outcomes and development conceptualization.

DEDICATION

To my parents

To my wife Hea Sung Moon,

To my daughter Inyoung Kim and my son Ethan Saiyoung Kim

ACKNOWLEDGEMENTS

Many people have supported my research that has led to this dissertation.

I would like to express my deepest gratitude and appreciation to my committee chair, Dr. Mark J. Clayton, and the committee members, Professor Geoffrey J. Booth, Dr. Valerian Miranda, and Dr. Wei Yan for their guidance and support through the course of this research.

Serving as my advisor, Dr. Clayton has helped me from the beginning of the study in course selection, decision of the research topic, implementation of the research plan, to dissertation writing. His insightful advice and broad experience in architectural research have enabled me to complete this research.

As my committee member, Professor Booth has encouraged me to investigate the application of my research in land development and enabled me to experiment research findings through the QNV-IV system development research. His intellectual inspiration and financial support have formed a solid foundation of this dissertation. I would like to thank to Dr. Miranda for his broad knowledge and intellectual advice that helped me to refine the research structure and the methodology in the early stage of this research. Dr. Yan has inspired me to study the architectural computing and software development. The informative guidance and the financial support from the NSF-PBIM research were also significant for this research.

My friends and colleagues in BIMSIM research group contributed the research. Their considerable fellowship made my life at Texas A&M University a great experience. In particular, Dr. Firas Al-Douri and his research inspired me to initiate this research. I would like thank to WoonSeong Jeong, James Haliburton, Francisco Farias, Saied Zarrinmehr, Duygu Yenerim, Mohammad Asl, Jawad Altabtabai, Chengde Wu, and Hyoungsub Kim

Finally, my parents, Yoenki Kim and Sangmin Park, provided continual encouragement and financial support. Special thanks to my wife Hea Sung Moon for her endless support for taking care of my daughter Inyoung and my son Ethan Saiyoung for love, patience, and sacrifices in many years of study. I dedicate this work to them in gratitude for their love, devotion, and support.

NOMENCLATURE

PURM	Parametric Urban Regulation Model
PUDM	Parametric Urban Design Model
PUDO	Parametric Urban Design Model Object
PUDA	Parametric Urban Design Application

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Overview

For the urban design stakeholders, regulations are becoming more difficult to incorporate into designs (Carmona, 2009; Hascic, 2006; Imrie and Street, 2009). As more prescriptive urban regulation approaches have been evolved to form places in the United States, development, interpretation, and implementation of the urban regulations have become more complex (Ben-Joseph, 2005; Stephenson, 2002; Talen, 2009). At the same time, policy makers are raising expectations for greater predictability of impact of the urban regulations on built environment with respect to sustainability for energy, food, health, and climate (Talen, 2012).

This research explored a conjecture that Parametric Urban Regulation Models (PURM) can express the form implications and the development performances of urban regulations prior to the regulation adoption. The development and testing of a software prototype have implied that PURM can achieve benefits in comparison to conventional methods of representing zoning codes. The software prototype has been developed by using the parametric modeling within Building Information Modeling (BIM) technology, the Application Programming Interface (API) in the BIM authoring tools, and the Object-Oriented Programming (OOP).

The research investigated 1) whether the key provisions of the urban regulations can be captured as the parametric BIM objects, 2) whether interrelationships among the regulation constraints can be represented as software algorithms in BIM, and 3) whether development performances can be deduced using the new urban regulation models in parametric BIM.

1.2 Definitions

In the context of urban regulations in the United States, various terms are used such as law, rules,

regulations, codes, ordinances, standards, guidelines, etc. Rules and regulations collectively formed codes such as zoning codes, which have affected urban pattern and form in diverse ways (Ben-Joseph, 2009). The ordinance often refers to municipal law, whereas law generally occurs at any scale of municipalities (Talen, 2013). Urban design standards and guidelines differ from codes in view of legal enforceability; codes tend to be regulatory, while standards and guidelines are often advisory (Talen, 2009).

Zoning has had the broadest effect on urban form and scale among urban regulations in the United States (Punter and Carmona 1987, Talen 2013). This research placed a higher priority on the zoning codes than other urban regulations to understand the context of urban regulations in the United States, to identify the common structure of urban regulations, and to establish the PURM development framework. In a similar manner, some contents in the literature review and case studies in the following chapters focused on the zoning codes in the United States.

Zoning have been employed as a regulation tool and various types have been applied in the United States such as Euclidean, Performance, Incentive, and Form-based Zoning. Euclidean zoning separates land uses into zoning districts and specifies permitted and excluded land uses to protect the neighborhood from undesirable land uses, establish public health, and prevent overcrowding and congestion. Euclidean zoning is the most prevalent zoning approach in the United States, while Performance and Incentive zoning have often been used by combining with the Euclidean zoning. In this research, zoning codes of Euclidean, Performance, and Incentive are categorized as conventional zoning codes.

Recently, form-based zoning is widely applied. Some organizations use the name “form-based codes” for proprietary code products, while the phrase is also used in research literature to describe a broad type of zoning codes that focus more upon the building form than on building

use. Hence, it is necessary to clarify the definition of the form-based code and the zoning codes in this research. The zoning codes and the form-based zoning codes are defined as follows:

- Form-based zoning codes and their various spellings such as Form-based code(s) and Form-based design code(s) are abbreviated as FBC in this research.
- The FBC in this research does not refer to the specific code product created by organizations such as Form-Based Code Institute or by private code creators such as Duany Plater-Zyberk & Company (DPZ)'s SmartCode.
- Zoning codes in this research refer to codes that regulate urban form and scale of public spaces, streetscapes, buildings, and parking spaces through design standards or guidelines.
- Whether they are labeled FBC or not, the zoning codes in this research may have similar formats, components, and constraints of the FBC.

Significant definitions about the Parametric Urban Regulation Models are as follows:

- *Parametric Urban Regulation Model (PURM)*: a new urban planning and design platform composed of Parametric Urban Design Model and Parametric Urban Design Applications.
- *Parametric Urban Design Model (PUDM)*: an urban district model in BIM composed of a series of Parametric Urban Design Model Objects. This corresponds to or expands the land use maps or the regulating plans of the urban regulations.
- *Parametric Urban Design Object (PUDO)*: geometrical representations of built environments shaped by the urban regulations such as street, open space, building, and parking.
- *Parametric Urban Design Application (PUDA)*: The custom application created in the

BIM API that (1) establishes the relationships among the PUDOs and (2) performs complex operations in urban design, which cannot be handled only by the PUDO.

Development performances of the urban design regulations are often broad, unforeseen, and unintended. Talen (2012) claimed that the urban regulations have affected the physical character of cities such as urban pattern, use, and form in explicit and direct ways. Barnett (2011) stated that the urban regulations have shaped urban form and building envelope such as street frontages and building height. In this research, I define development performances of urban regulations as form implications of building envelope as well as economic and environmental performances.

This research was conducted using 1) a literature review to identify the urban regulation history in the United States, 2) an analysis of a sample of recent zoning codes, 3) creation of the PUDM in parametric BIM, 4) prototyping the PUDA, and 5) experimenting and evaluating the PURM.

1.3 Problems and Challenges

1.3.1 Challenges in use of the urban regulations

In the United States, alternative urban regulation applications such as Form-based zoning have been created to substitute or support the conventional regulation systems (Barnett, 2011; Forsyth, 2003). Recent zoning code reforms have been promoted under the question of how new developments can minimize ecological impact, achieve sustainability, and address sense of place (Barnett, 2011; Walters, 2007). However, there are still inherent and unresolved issues in creating and using the urban regulations.

1.3.1.1 Complex to create and difficult to use

Many recent urban regulations have placed a higher priority on controlling urban form and scale in comparison to conventional regulations such as Euclidean zoning regulations that focus on

density and use of buildings (Barnett, 2011; Talen, 2012). FBC uses prescriptive provisions that determine geometrical attributes of built environments. For better understanding of the prescriptive provisions, graphical standards and figures are included in the codes. However, incorporating such regulations into design is still time consuming and error-prone for the code users due to the several reasons.

First, a regulation provision is explicitly or implicitly related to other provisions; often multiple provisions control urban form and scale of one element of built environment. For instance, the capacity of parking spaces is frequently determined by the provisions controlling the building type and scale. Disposition of parking spaces is related to the building location and the context of subdivision, but multiple provisions such as setback and front façade location requirements constraint the building location. The subdivision regulations affects the subdivision geometry that affects the building design. In sum, these regulation provisions are related to each other.

On the other hand, the provisions are expressed in a textual document with broad applicability, complex relationships across the provisions make it difficult to locate relevant or required information. Further, to understand form implications of the urban regulations, the users need to superimpose the collected information from multiple provisions onto the particular project sites. The wide range of the site-specific constraints makes this overlay process more complex in that urban form and scale such as disposition and height are related to the site topologies (Kim, Clayton, and Yan, 2011).

Although the urban regulations often provide a user manual section to explain how to use the components and the provisions, understanding form implications from the associated multiple provisions and checking compliance of a design with the regulations are still challenging for planners and designers. Consequently, urban planning and design review can delay both the

planning process as planners create and amend the regulations and the design process as designers assure the regulation compliance of their design (Kim, Clayton, and Yan, 2013).

The diverse types of information in urban design increase the complexity in using the urban regulations. Typical data sets for urban design include Geographic Information System (GIS) data for the environmental constraints, textual documents for the regulatory constraints, Computer-aided Design (CAD) data for subdivision and platting information, and vector images for the master plans or the development plans. These data types are often in different scale, detail, and formats, which can impede the urban design process.

1.3.1.2 Complex interpretation of the urban regulation provisions

Although there are varying naming conventions and structures, recent urban regulations often include the urban plan such as a master plan and a zoning map as well as a series of regulation standards for open space, streetscape, building, parking, and architectural design. The regulation users must collect the requirements from the entire document and then overlay collected information to the urban plan (Figure 1). The typical overlay process is as follows:

- Define the classification types such as land use or the transect codes from the urban plan.
- Define the disposition requirements of buildings and the parking structures from the urban plan.
- Define the type of streetscape, public space, building, and parking from the urban plan.
- Locate and retrieve each standard and then identify the requirements and the types from the regulation standards.
- Overlay collected information into the site in the urban plan.

The concept of the overlay process traces back to the “City as Layer”, an urban design strategy

proposed by O. M. Ungers (Ungers and Vieths, 1999). His notion is that the modern cities have complex structures and consist of a series of superimposed layers such as transportation, parks, water, and buildings. The overlaying approach has been accepted in urban planning and design to make the complex structure effective. This concept is reflected not only in the GIS data structure but also in the spatial operation processes of the GIS tools.

However, the overlay process may not be applicable to the recent urban regulation implementation without the complex interpretation process to form the three dimensional geometry. For example, to overlay the requirements for the vertical configurations and the horizontal dispositions, it would be required to use complex operations that can handle 3D urban geometry. The use of 2D regulatory tools often result unforeseen consequences when applied to the multi-dimensional built environment, which constitutes a major flaw in zoning as a regulatory tool.

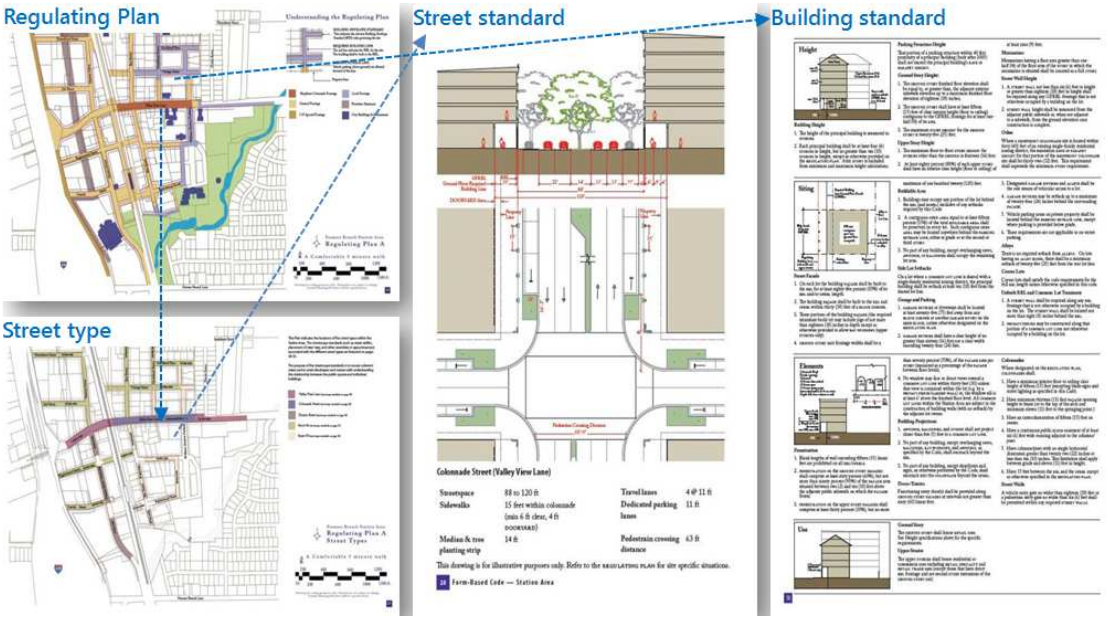


Figure 1: An overlay process among the regulation components. Adapted from the Station Area Development Code (The City of Farmers Branch, 2014).

1.3.1.3 Assessing the impact of urban regulation provisions

Owing to growing concerns on sustainability, urban regulation reformers have established new goals and objectives, amended the regulation provisions, and often enforce or recommend to use a rating system of environmental foot print of urban development such as LEED for Neighborhood Development (LEED-ND). However, the evaluation through such rating systems can be performed by the individual developers or designers during the building design phase. The assessments on the effect of the urban regulations on the built environment have not been carried out in the early stage of the code making process.

Development capacities and energy performances are significant indicators for predicting environmental footprints for the resource managements (Fischer and Guy, 2009; Lang, 1994; Punter, 1997). The prescriptive urban regulations such as FBC are less rigid in limiting density than the conventional zoning. Often, the target population or the expected development capacities are not clearly expressed in the zoning codes, the comprehensive plans, or the master plans. As a result, measuring development capacities and estimating performances of the development schemes are often carried out not in the code making process by code makers but in the building design process by developers.

Such evaluations on the development performances may be certain, but it is too late to be reflected on the zoning code making process. Immediate evaluations during the development of urban regulations would be critical to ensure regulation reforms will or can contribute to the more sustainable built environments.

1.3.2 Design oriented urban design tools

In urban planning and design, there is recognition of the significant role of the computer tools to visualize and analyze the built environments (Brail, 2008). Al-Douri (2006) explained that

computer-based urban modeling allowed users to describe existing and future cities without the traditional physical urban models. In response to the needs for the better computational modeling platforms, various computer tools have been developed, but they have paid attention to the analysis of the environment, the fast urban model generation, as well as the realistic visualization.

GIS is used to represent essential data in the urban planning and the design process, such as environmental, geographic, demographic, network, structure, and aerial photos (Brail, 2008). Limited regulation information such as land-use and zoning is also often included in the GIS data sets. ESRI ArcGIS, a widely adopted GIS software system, provides diverse spatial operations such as intersect, union, identify, clip, and erase geometry as well as various analysis features such as surface, distance, density, solar, hydrologic, and statistical analysis.

To extend GIS beyond its paradigm of 2D overlays, ESRI acquired CityEngine, a 3D modeling application for urban environments (ESRI, 2014). This was initially developed by ETH Zurich and commercialized by Procedural in Switzerland. The tool focuses on fast 3D modeling and realistic visualization based on the procedural algorithm and GIS data. Recently, ESRI added an interface for editing the regulation parameters, but limited parameters can be applied to the models.

Many software companies have created new tools or modified the tools for fast urban modeling and visualization using GIS data, such as Autodesk Civil (Autodesk, Inc., 2014), Infraworks (Autodesk, Inc., 2014), Bentley Map (Bentley Systems, Inc., 2011), and CityCad (Holistic City Limited, 2014). These tools provide powerful features for robust and quick visualization (Gil and Montenegro, 2010), but limited regulation information can be represented. They have limited features for the parametric modeling and programming interface.

1.3.3 The potential of urban regulation modeling in parametric BIM

BIM has been widely accepted in the Architecture, Engineering, Construction, and Owner-Operations (AECO) industry. BIM software environments couple 3D modeling with parametric form generation and rich semantics, which can express design concepts through real-time design changes (Eastman, Teicholz, Sacks, and Liston 2008; Sacks, Eastman, and Lee 2004). Parametric modeling enables objects to express geometry through parameters. The relationships among parameters can be defined using formulas. As objects store information as parameters, elements of objects can be regenerated according to parameter change. The 3D modeling provides for the ability to generate accurate visualizations of urban and building form and simulate the effects of natural and artificial lighting. In addition, BIM platforms provide an API, which can facilitate the creation of add-in programs with easy access to building information and ability to customize special operations to perform complex tasks. For these reasons, BIM is becoming the standard environment for building design, construction, and facility management.

Owing to these capabilities, BIM has potential to represent not only the impact of urban regulation provisions on human settlements but also the relationships among the provisions. Furthermore, the API enables the development of analysis and simulation capabilities that can show the performances of urban design schemes. The parametric modeling capability in BIM may enable an urban model to rapidly present multiple development scenarios. The visualization capabilities may enhance the understanding of our future built environment. The analytical features in BIM or via BIM may enable performance assessments of the future development. Testing the scenarios, visualization of the resultant built environment, and the real-time assessment of the urban regulations may appeal to various expert and inexpert stakeholders and prove effective in supporting decisions about the development of the urban regulations.

However, the textual statements in the regulation provisions significantly differ in semantics and ontology of the objects and user interfaces of BIM software. Therefore, to take advantage of the capabilities of BIM it is significant to investigate new methods and processes to represent the textual regulation provisions in BIM.

1.4 Goals, Objectives, and Significances

The major goal is to enable stakeholders to devise and use the urban regulations effectively by understanding the intention and the development performances of the urban regulations. Toward this goal, I will develop the PURM, a new computational approach to present urban regulations, by using parametric BIM technologies. The major objectives are as follows:

- Investigate a new method for modeling urban regulations by using parametric BIM technology, so that the new model can present the intention and the development performances of the urban regulations.
- Investigate the prototypes to understand to what extent it can capture the textual urban regulation provisions as algorithms and ontologies of parametric BIM.

The significance of this research includes the following:

- The research will potentially provide a new approach to represent textual regulation constraints of the urban regulations with computation models in parametric BIM.
- The proposed system is expected to enable municipal officials who are responsible for the development of urban regulations to envision the future development resulting from the regulations prior to their adoption.
- The proposed system has potential to enable the code makers to analyze performances of a series of urban regulation schemes, which can inform the critical factors of the

regulation provisions.

- The proposed system will potentially alleviate the difficulties in urban regulation interpretation, which can reduce the time and effort in the regulation compliance checking or the planning review processes.

1.5 Research Questions and Propositions

This research will address the following primary question:

- Can parametric modeling and OOP in BIM parameterize the textual regulation provisions, capture the complex constraints in the provisions, and express form implications of the urban regulations with algorithms and ontologies in BIM?

To address the primary questions, this research will answer the following secondary questions:

- Can the important urban regulation provisions be modeled in the PURM?
- Can important relationships among the regulation provisions be modeled in the PURM?
- What are the challenges and potential of the parametric BIM technology in the PURM development?

The research proposition is as follows:

- If the parametric BIM technology can represent a part of the urban regulation constraints controlling urban form and scale and if the parametric BIM technology can visualize a part of the intention and the performances of urban regulations, then the PURM can enable stakeholders to devise and use the urban regulations effectively.

The sub propositions are as follows:

- A PZRM based on parametric BIM can represent common, typical, and important provisions of urban regulations.
- A PZRM based on parametric BIM can enable tests of consequences of candidate urban regulation provisions before adoption of urban regulations.
- A PZRM based on parametric BIM can reduce ambiguity in urban regulation interpretation during the design phase.

1.6 Research Strategy

The main research phases are (1) review of literature on urban regulations, (2) analysis of recent zoning code samples, (3) parameterization of urban regulations with PUDM, (4) development of PUDA, and (5) experiment and evaluation (Figure 2).

In each phase, I will employ mixed research methods consisting of a literature review, case studies, prototyping, simulation, and model-based reasoning. The following sections describe major research criteria, research methods, data, and tools of each phase.

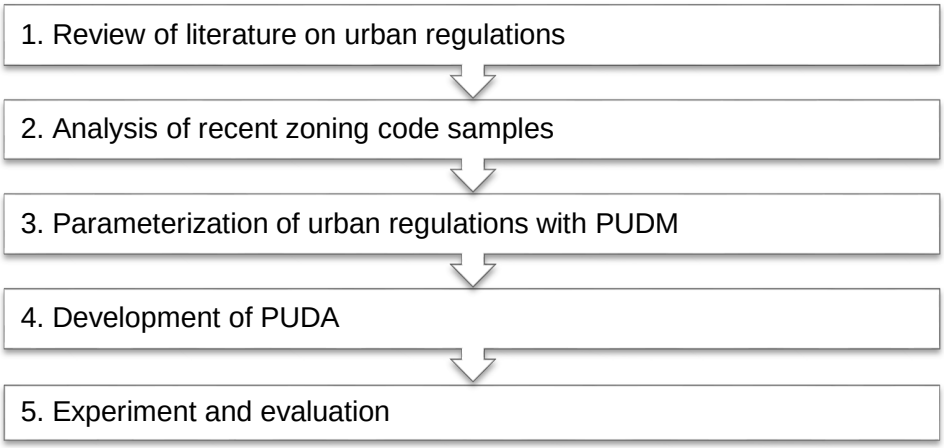


Figure 2. Main research phases

1.6.1 Review of literature on urban regulations

I will conduct a literature review on urban regulations in the United States to establish the context for urban regulation modeling research and identify appropriate technologies to advance how urban regulations are implemented: Three topics are addressed:

- A review of the urban regulation history identifies (1) concerns and the strategies commonly addressed in urban regulations, (2) the distinctive approaches of different types of recent zoning codes, as well as (3) complexities on the urban regulation implementation.
- A review on urban regulation modeling traces the accomplishments and the unresolved issues in the computational urban regulation modeling.
- A review on the BIM identifies the capabilities of BIM in the AECO industry, the status of the current urban design tools, and the potentials of the parametric BIM technology for urban regulation modeling.

1.6.2 Analysis of recent zoning code samples

By using a sample of contemporary zoning codes, I will review following criteria: (1) the common types of structures, components, and provisions of the zoning codes, (2) the chapters, section, and provisions controlling form and scale, (3) the main elements of built environment governed by the zoning codes, as well as (4) the interrelationships across the code components and the provisions for controlling scale and form.

I will conduct case studies on the following zoning codes:

- Heart of Peoria Form Districts (City of Peoria Illinois, 2014).
- Near Southside Development Standards and Guidelines (Fort Worth South, Inc., 2013).

- Downtown Specific Plan (City of Ventura, 2011).
- Station Area Development Code (The City of Farmers Branch, 2013).

The selected code samples have clear project boundaries for neighborhood scale development. Zoning codes in city, county, or state scale are excluded. They place higher priority on controlling urban form and scale with specific provisions such as Required Building Line (RBL) or Built-to Line (BTL). In addition, they use transect classification systems for diverse scale, form, and architectural characteristics.

The land use or the zoning maps will be collected in digital file formats such as GIS data or CAD files. To use GIS data, ArcGIS will be used. ArcGIS allows users to manage both spatial and attribute data with the convenient operating functions (Göçmen and Ventura, 2010; Lunetta and Lyon, 2004), but additional data processing such as a file conversion or a data translation is required to use GIS data in the BIM platform.

The findings from the case study are the types of PUDOs, a set of parameters of the PUDOs, and an object hierarchy among the PUDOs. Such implications will be the foundation for the prototype development in the next two phases.

1.6.3 Parameterization of urban regulations with PUDM

In this phase, I will develop the PUDO and the PUDM in BIM. The major tasks are investigating how to establish an object hierarchy among the PUDOs, how to map the textual regulation constraints into the PUDO, and how to compose the geometry of the PUDO in BIM. The PUDO will be used in prototyping the PUDA in the next phase.

For urban regulation modeling in parametric BIM, I will utilize Revit (Autodesk, Inc., 2014), a widely used BIM authoring tool. The parametric modeling features in Revit enable a BIM object

to store its attributes as parameters and to transform object geometry by changing the parameters. Owing to the comprehensive building component library in Revit, the Parametric BIM Object can be used to rapidly create building envelope by using building components. Further, the data structure of the BIM object in Revit can be accessed from a customized application created using the Revit API.

1.6.4 Development of PUDA

In this phase I will develop software prototypes of the PUDA that establish relationships among the PUDOs and their parameters. Three scopes of applications development are: (1) representing relationships among the PUDOs and their provisions; (2) performing the complex operations in the overlay process of the multiple PUDOs; and (3) enhancing the modeling process of the PUDM.

The BIM authoring tools provide software developers with various APIs such as Revit's .Net based API in C# and VB.NET, MicroStation's macros in C++, C#, and VB.NET, as well as Graphisoft ArchiCAD's API in C and C++. API eases complexity in software development by providing specifications of data structures, object classes, and variables. BIM API allows assessing information of the BIM objects, creating and editing the BIM objects, creating a custom Graphical User Interface (GUI), and performing various analyses using BIM.

In software programming, OOP enables source codes to be modular and reusable. A class in OOP is a data type that can describe an object representation, so individual objects are created using the classes. A class determines object's common characteristics, attributes, and behaviors (Tello, 1989). In prototyping, I will use Revit API as software development specifications, Microsoft C# as an OOP language, and Microsoft Visual Studio as an Integrated Development Environment (IDE). In addition, existing open source libraries in C# will be utilized to read and

write an external database.

1.6.5 Experiment and evaluation

Lastly, I will demonstrate how the PUDM and the PUDA are integrated into the PURM. To assess the achievement of the research objectives, an experiment will be carried out through a test case with an existing zoning code, Form-based Code Station Area of Farmers Branch that was adopted to regulate development near the DART station located in north of Dallas.

In the experiment, I will assess whether the application development scopes are addressed. A set of the PUDOs will be built and assembled to create the PUDM. Then I will simulate how the PUDAs interact with the PUDM to address difficulties in use of the urban regulations. Lastly, I will describe the observation results on how the economic and environmental analyses can be conducted in conjunction with the analysis capabilities of the PURM and the native features of the BIM authoring tools.

1.7 Expected Outcomes

This research is an investigation of the PURM as a new platform for the urban regulation development, conceptualization, and regulation compliance checking. The PURM expresses urban regulations as 3d geometry of BIM objects and parametric relationships in a BIM environment.

PURM, as a new urban regulation models in BIM, is expected to be used by the various sectors such as planning departments, code makers and urban designers, building designers and developers, as well as citizens. For instance, the planners will use this model as a regulation development and analysis system to envision the development and economic performances prior to the code adoption. The urban regulation makers and the urban designers will use the system in

regulation creation and master plan design. The designers and the developers will use 3D urban regulation model to enhance understanding on the regulation provisions. The citizens will be able to understand the future growth of their neighborhood.

Once the new urban regulation models are expressed in parametric BIM, various special tasks will be able to be conducted. For instance, the regulation allowances and the designed building information can be stored in the same BIM database. The urban regulation models in BIM may be used in the code compliance checking process for individual developments. The models will be able to provide a complete range of information about the urban regulation constraints, subdivision topology, and building design, which can be used in the various research and analysis. A range of up-to-date analysis tools supporting BIM will be applicable to the urban regulation models, enabling stakeholders to perform environmental performance analyses for wind, solar, daylighting, and energy consumption, etc.

In the long run, the approach and findings may ease the complexities in the urban regulation making process for municipal officers and regulation developers and in the zoning code interpretation process for urban designers and architects, which may facilitate a more integrated process among urban planning, urban design, and building design. Eventually, the research contributions may even remediate some of the flaws inherent in in zoning as a regulatory tool.

2 REVIEW OF LITERATURE ON URBAN REGULATIONS

In this chapter, I will conduct a literature review of urban regulations to establish the context of urban regulation modeling research. The major scopes are urban regulations and zoning in the United States, computer-based urban design, and the potential of the BIM technologies for urban regulation modeling.

A review on urban regulations and zoning in the United States identifies (1) background of urban regulations and zoning, (2) characteristics of the recent zoning regulations, and (3) challenges in the use of recent zoning regulations. A review on computer-based urban design identifies (1) the status of the urban design tools and (2) parametric urban design as a theoretical paradigm. A review on the potential of BIM for urban regulation modeling identifies (1) BIM capabilities in the AECO industry and (2) Parametric modeling and OOP in BIM for the urban regulation modeling research.

2.1 Urban Regulations and Zoning in the United States

Various types of urban regulations have been created upon particular concern with various strategies, but zoning has played a significant role in shaping the built environment in the United States.

2.1.1 Shaping urban form with urban regulations

In shaping urban form with urban regulations, the United States followed Europe. In the 17th century Europe, building ordinances and regulations affected urban form and scale (Hall, 2002). In contrast, some restrictions controlled building materials, but regulations rarely influenced urban from in the United States at that time (Ben-Joseph, 2009). For instance, New York's Tenement House Act of 1867 was adopted to protect public health and private investment, but

this regulation was mainly applied to individual housing developments (Talen, 2012).

In the 17th and the 18th century, nuisance laws regulated undesirable activities in the city, which became the basis of zoning (Ben-Joseph 2005, Check). The 1692 regulation in the Common wealth of Massachusetts in 1962 designated the allowed location of the slaughterhouses in the city (Melosi, 1999). New York's 1703 law designated that certain types of industries could not be located within a half mile of the city hall (Hall, 2002).

Deed restrictions controlled building form and use before zoning, but the restrictions were more common in private community developments (Punter, 1997). Deed restrictions aimed to preserve the residential character and aesthetic uniformity by limiting commercial and industrial facilities in residential area. The regulations often limited not only a range of housing price within the subdivision to control the economic class of the neighborhood but also setback from the street to maintain residential density (Talen, 2012).

2.1.2 Background of the American zoning

No urban regulation has had more impact than zoning in the United States (Talen, 2012). Zoning was created to remedy the negative aspects of the industrial city, but the origin of zoning can be traced to the ancient Greeks and Romans regulations. In the Greeks regulations, the cities were zoned to separate residential areas from civic and religious functions. In Romans laws, industry and residences are separated as well as commercial uses were allowed adjacent to the main corridor (Ben-Joseph, 2005). The zoning approach is also found in London and France. In the 17th century London, some industries such as leather tanning plants required to be built outside of the city center and shops were not allowed on main public squares and on the main streets. In the 19th century France, unhealthy industries should be kept out of cities (Reynard, 2002).

The zoning regulation approach was further developed by the German engineers Reinhard Baumeister and Franz Adicks at a meeting of the German Architectural and Engineering Societies in 1874 (Ben-Joseph, 2005). In Baumeister's book, *Town Expansions Considered with Respect to Technology, Building Code, and Economy* (1876), the regulation approach was called as zoning and the origin was traced by the regulations in the 19th century France. He created two zones for the city and the suburb and then specified building bulk regulations for building height, setbacks, and the lot area, which influenced German cities in 1890s. Later the German zoning approach influenced in the 1909 Town Planning Act and the Garden City movement in Britain (Baumeister, 1874).

The Garden City movement advanced in England in the late 1880s (Hardy, 2005). The Garden City movement proponent, Ebenezer Howard, claimed that suburban settlements need to be built with ample space, well-built clean houses, and abundant garden space. The monograph written by Clarence Perry, "Neighborhood Unit", explained the appropriate scale of suburban settlement. He proposed a basic plan for the ideal neighborhood size with major streets, a school, and neighborhood shops at intersections, which served as coding templates in recent decades (Kohr, 2004; Mehta, 2006).

City Beautiful is the pioneering systematic movement by American architects and policy-makers to address issues of city form in the early 20th century (Barnett, 2011; Hall, 2002). The City Beautiful movement proponents believed that a well-designed urban plan and the monumental building design can enrich civic character. The City Beautiful movement proponents were influenced by the Beaux Arts architecture in Paris. The McMillan Plan for Washington restored the primacy of L'Enfant's plan, which became the foundation of monumental government building plans during the next forty years. Daniel Burnham's 1909 Plan for Chicago placed an

overlay of Parisian boulevards on the existing grid street system, which connected the lakefront and a series of monumental parks. The spirit and the approach of shaping urban form in the City Beautiful movement inspired many urban reformers and the new urbanists (Barnett, 2011; Kohr, 2004.).

The Congrès Internationaux d'Architecture Moderne (CIAM) was a series of international conferences founded in Switzerland in 1928 by a group of modernist European architects. One of the most forceful proponents was Le Corbusier (Barnett, 2011; Hall, 2002; Lang, 1994). They emphasized architectural coherence, the optimal spacing of buildings, the best block sizes, and appropriate street typologies. In 1935 Le Corbusier planned Manhattan as a superblock city with tall buildings, green space, and high-speed streets. Housing in towers separated by open space became a mainstream architectural concept which was advocated by the CIAM. They reordered the city structure by function. Housing and work were separated, and open space and automobile transportation were allocated to connect the separated functions. This concept influenced many subsidized housing units for low-income communities in the United States before World War II (Hall, 2002).

These precursory urban design movements influenced the approach of shaping built environments by urban regulations in the United States.

2.1.3 1916 New York comprehensive zoning

During the 18th and 19th centuries in the United States, a few cities adopted sanitation laws for public health (Ben-Joseph, 2009; Hall, 2002). As American urbanization rapidly increased in the 19th century, business and industrialization were centered on the cities. Factories were built in urban areas, the communities were overcrowded, and rich inhabitants moved outside the urban area.

Modernist city design was begun to create a collective society where everyone would have housing for the minimum standards for sanitation, light, and air (Barnett, 2011). Most of the old urban areas were reshaped to eliminate slums and factories from communities. During this time, the codes were adopted to cope with the early urbanization and American zoning was designed to protect existing neighborhood from inappropriate developments in the early 20th century. Zoning was considered as a primitive system that could keep residences away from the noisy and dirty factories and could protect neighborhoods from tall buildings.

In 1916 New York City adopted the new zoning code to address both urban renewal and neighborhood preservation, which is the first modern comprehensive zoning ordinance in the United States. A series of spatial districts, including Broadway, Fifth Avenue, and the Lincoln Square Special District, were designated to spur public and private investigation and to preserve traditional fabrics. The Lincoln Square special zoning district used build-to lines to hold the street frontage (Barnett, 2011; Ben-Joseph, 2009; Talen, 2009). Floor Area Ratio (FAR) of these zoning codes has been widely used to control a building area and the potential population.

The New York zoning code used the height and setback controls that were previously used in the Parisian city design (Barnett, 2011). The height limit in Paris is an example of the powerful urban regulations in the pre-industrial cities, which made a uniform street frontage in Paris since the mid-eighteenth century. Buildings were required to be recessed from the property line and the maximum building height can be calculated according to the applied setback distance. The regulations for the Lincoln Square special zoning districts are regarded as a foundation of FBC.

Zoning was initially believed as the ideal tool, preventing over intensive development (Bassett, 1922), a spread-out of the city (Bettman, 1925), an excessive segregation, and the unconscious developments of neighborhood context (Hall, 2002). Zoning led citizens' participation in

planning in a belief that zoning can improve the architectural quality (Baker, 1927), alleviate poverty, decrease housing costs (Nettlefold, 1914), and increase productivity and profits (Adams, 1935).

The foundation of land use and zoning regulations in the United States was shaped by Standard State Enabling Act (SSZEA) in 1921 and Standard City Planning Enabling Act (SCPEA) in 1927 (Talen, 2012). In 1921 Secretary of Commerce Herbert Hoover formed an Advisory Committee on City Planning and Zoning and created the draft of planning and zoning statutes. In "A Zoning Primer", he stated that zoning is an efficient approach to city building to protect property and health as well as reduce the cost of living (Hoover, 1926). Owing to these acts, municipalities were able to create master plans and comprehensive zoning plans. Consequently, zoning expanded rapidly. The American cities having zoning increased from 62 to 456 from 1924 to 1926 and about eight hundred zoning ordinances were adopted in 1929 (Talen, 2012).

2.1.4 Zoning regulation types in the United States

In the United States, a wide range of zoning regulations has been evolved over the years as political, social, and economic priorities have shifted (Holm, 2006). Various theories of urban planning and design have influenced zoning approaches. Most zoning regulations changed built environment and the undesirable impact of the zoning regulations were immediately visible (Talen, 2012). Among numerous zoning types, the representative and distinctive zoning approaches are Euclidean, Performance, Incentive, and Form-based (Barnett, 2011).

Euclidean zoning is one of the most prevalent zoning methods in the United States. When it was introduced in the town of Euclid, Ohio, local land owners challenged this city ordinance because the use of private property may be restricted under the zoning. The case was upheld by the U.S. Supreme Court (in Village of Euclid v. Ambler Realty Co.) and decided in 1926. The goals are

establishing orderly growth by preventing overcrowding, relieving congestion, and separating incompatible uses (Ben-Joseph, 2005). To do so, this regulates land development using land use controls and dimensional standards. Euclidean zoning prescribes the allowed land use within each district and then excludes other types of uses. Basic classifications of land uses include residential, commercial, industrial, institutional, and recreational. Developments in each land use shall comply with the dimensional standards that regulate setbacks, height limits, lot coverage limits, lot size limits, etc.

Performance zoning regulates land developments for environmental protection by using performance standards on traffic flow, density, noise, air, light, etc. It is also called as Effective-based zoning. In this zoning, grading systems often administrate land development. Under the performance compliance, any building forms can be built, which allows a level of flexibility in design and administration (Ben-Joseph, 2005). However, it has not been widely adopted in the United States compared to Euclidean zoning, while it is used in hybrid approach by combing it with Euclidean zoning (Barnett, 2011).

Incentive zoning was implemented in Chicago and New York first. Incentive zoning regulates developments to meet established goals using a reward-based system. In general, a base level of limitations is prescribed and a list of incentive criteria and reward scales are provided. For instance, providing affordable housing on site endows height limit bonuses. Incentive zoning requires a series of revisions to maintain balance between incentive magnitude and the value given to developers (Hascic, 2006).

FBC is a zoning regulation approach that aims to achieve a specific urban form rather than building functions and bulks. U.S. Environmental Protection Agency (EPA) defined form-based codes is a type of zoning codes that outlines specific urban form rather than zoning by use

(2011). EPA differentiated FBC with design guidelines and design standards in view of significant enforceability. Talen (2009) explained that FBC is a linage of zoning codes rather than design guidelines or standards. She defined the attributes of FBC as significant enforceability, the prescriptive regulations, and the production of urban form of urbanism.

FBC has been widely applied in the United States after the FBC in Seaside in Florida, Celebration in Florida, and Cotton District in Mississippi. Initially, the code was called as FBC and later DPZ named their zoning code products as SmartCode. There are various modified zoning codes similar to FBC. Even though zoning codes have the attributes and the approaches similar to FBC, such codes are not always classified as FBC. Moreover, some zoning codes and FBC are not clearly distinctive each other.

Ben-Joseph (2009) stated that controlling urban form with the integrated regulations and the design codes is a continuing phenomenon in the urban regulation reform history. Others claimed that continuing urban regulation reform has been influenced by precursory theories and strategies of City Beautiful, CIAM, Garden City, and the late 19th and the early 20th century's American zoning (Kohr, 2004; Mehta, 2006).

2.2 Characteristics of the Recent Zoning Regulations

After Duany Plater-Zyberk and Company (DPZ) created FBC for Seaside, Florida, FBC became increasingly popular and the regulation templates such as DPZ's SmartCode were created for zoning regulation reform. Many municipalities modified the regulations by using either the previously reformed regulations or the regulation templates. Consequently, similar structures and approaches with FBC are discovered in many zoning regulations in the United States. The following characteristics are often discovered either from FBC or the recently reformed urban regulations in the United States.

2.2.1 Prescriptive provisions controlling urban form

FBC places higher priority in controlling urban form such as the typology of block, street, open space, and building envelope (Ben-Joseph, 2005; Parolek, Parolek, and Crawford, 2008). FBC often controls building disposition with prescriptive provisions. For instance, Built-To-Line (BTL) and Required Building Line (RBL) designate the location of building façade. In many cases, setback lines for building and parking structure are defined in the Regulating map. For building height, ceiling height or clearstory height are often defined. Such provisions are regarded as less flexible than provisions of Euclidean zoning codes.

2.2.2 Regulating plan

In FBC, the regulating plan is the most significant component (Parolek et al., 2008). The regulating plan designates the appropriate form and scale of developments rather than defines allowable land-use types as Euclidean zoning codes do. In the regulating plan, each property is ruled by transect codes, which shows actual development requirements such as building location, setbacks, or building heights. The term of the regulating plan was frequently used in FBC, but various type of urban regulations either use the term or has a very similar plan to FBC's regulating plan.

2.2.3 Transect

Transect is an urban planning model that defines the hierarchical development scale from sparse suburban to dense urban cores (Forsyth, 2003; Hascic, 2006; Ligmann-Zielinska, 2008; Stephenson, 2002). The character of each transect zone is defined by the degree of density, open space, and urban form. The main components of FBC are the Regulating Plan having Transect Codes and a series of standards for public space, streetscape, building, and parking. Required regulation information is located in these components and users need to locate all constraints and

requirements from those documents (Parolek et al., 2008).

2.2.4 Graphical standards

FBC provides many illustrations and drawings to prescribe urban form such as building location and volume. Graphical standards present the relationship between building facades and the public realm, the form and mass of buildings, as well as the scale and types of streets and blocks. In general, graphical standards of FBC include public space standards, streetscape standards, building form standards, parking standards (Parolek et al., 2008). The graphical standards are easy-to understand compared to the text-based provisions, but they depict generic cases such as rectangular subdivisions or straight streets. It is still difficult to apply the graphical standards to certain sites having complex topology.

2.3 Critiques and Challenges of the Zoning Regulations

In this section, the major criticisms of the zoning regulations are described. Then challenges and difficulties in use of the recent zoning regulations are presented, which will be focused in this research.

2.3.1 Criticisms and limitations of the zoning regulations

Criticisms of zoning are broad. One of the criticisms is that the zoning regulations caused segregation of building use and size (Ben-Joseph, 2005). In the early 20th century American cities, various size and use of buildings were mixed such as houses, apartments, grocery stores, restaurants, etc. By the acts of SSZEA in 1921 and SCPEA in 1927, the residential-only neighborhoods with single family houses were legalized. As zoning regulations were spread out throughout the municipalities, houses were separated from apartments. The residential districts were filled with the houses having similar size, type, and price. Frequently, other uses were not allowed within the residential districts, so a commercial strip was built along the main corridors

(Barnett, 2011). Studies presented that zoning regulations in the United States have influenced segregation of income and race, urban sprawl, traffic congestion, and spatial mismatch between jobs and house (Ihlanfeldt, 2004).

The other criticism is that the zoning regulation is too simplified to regulate the complex city systems. Studies on the complex systems of the city were begun in the 1960s based upon criticism on modern planning in the 1950s and 1960s (Barnett, 2011). In "Death and Life of Great American Cities (1961)", Jane Jacobs claimed that the separation of uses by zoning regulations would destroy economics and activities in communities. She identified the city as complex layers and proposed four generators of diversity; 1) mixed uses, 2) walkable block size, 3) variety of building age and condition, and (4) higher density (Jacobs, 1992).

The City as Layer is an urban design strategy, systematized by O.M. Ungers (Ungers and Vieths, 1999). His notion is that modern cities are complex structures and the overlaying approach can make this complexity apparent. Cities were understood as a series of the superimposed layers, such as transportation, supply, services, parks, water, and buildings. In the Potsdamer Platz and the Leipziger Platz projects (1991), he presented the overlaying strategy with site boundary, axes, block structures, existing buildings, existing green area, and existing roads. His overlaying method has been widely applied in current urban design practices. In addition, the use of the superimposed layers is one of the key approaches in the regulation modeling of this research.

On the other hand, lack of environmental consciousness in the zoning regulations was criticized.

Design with Nature, written by Ian McHarg in 1969, is a pilot study on the appropriate development form for natural environments. McHarg emphasized the significance of cooperation among planning, design, and the science of ecology in community design (Steiner, 2004). He

argued that local zoning regulations mapped large areas with little concerns about the ecology and natural environments. For example, property lines and zoning boundaries often cut across the least suitable landscape for buildings, which could cause environmental problems. He asserted that civil engineers, planners, designers, and developers should pay more precise attention to the environmental impact to resolve these issues (McHarg, 1992).

Transect is a planning strategy that defines the hierarchical development scale from sparse suburban to dense urban cores (Barnett, 2011; Ben-Joseph, 2005; Stephenson, 2002). The original concept of transect was proposed by Patrick Geddes to understand many symbiotic elements within a range of habitats. In the Valley Section research, he divided habitats into six transect zones according to the distribution of water, soil, vegetation, etc.

The idea of transect classification is applied in McHarg's Woodlands project, an incorporated community development in Montgomery County, Texas. These environmental consciousness in urban planning and design influenced the transect zoning to address with dwindling energy resources (Stephenson, 2002). In transect zoning, the character of each transect is defined by the degree of density, open space, and urban form. The approach of transect will be applied in the urban regulation modeling in this research.

2.3.2 Challenges in use of the recent zoning regulations

Recent zoning regulations often use prescriptive provisions to control topologies of open space, street scape, building, and other physical design elements of built environment (Ben-Joseph, 2009). In general, graphical standards for open space, streetscape, building, and parking are used and user manual sections are included in the code document. However, studies state that use of the recent zoning regulations is still complex (Carmona, 2009; Imrie et al., 2009).

2.3.2.1 Complex overlay process

In many recent zoning regulations, the regulatory requirements are dispersed in the diverse code components and provisions. Regulating plan is the key component of the recent zoning codes, which is similar to the land-use map in Euclidean zoning codes. The regulating plan designates the type of regulation standards of open space, streetscape, building, and parking. Detailed provisions are defined in the regulation standards respectively (Parolek et al., 2008). After users locate all requirements from the regulation standards and the regulating plan, they need to apply collected requirements into design. While there are varying naming conventions in the zoning code components, the general sequence in using zoning regulations is as follows.

(1) Define the type of public space, streetscape, building, parking standards from the regulating plan.
(2) Define disposition information of the building and the parking structure from the regulating plan.
(3) Define the type of architectural standards according to the building standards.
(4) Overlay defined information into design.

In this research, the above process is named as the overlay process. Figure 3 presents the overlay process of the existing zoning code for the Farmers Branch station area development. In the next chapter, I will describe the overlay process of the selected zoning code samples. In many cases, the zoning codes include a section for the user manual to explain the overlay process, but the process is not always obvious for designers (Kim et al., 2011).

2.3.2.2 Interrelationships among the regulation provisions

Many provisions in the zoning regulations control urban geometry, and they are associated with others. For instance, the building and the parking structure size are correlated each other.

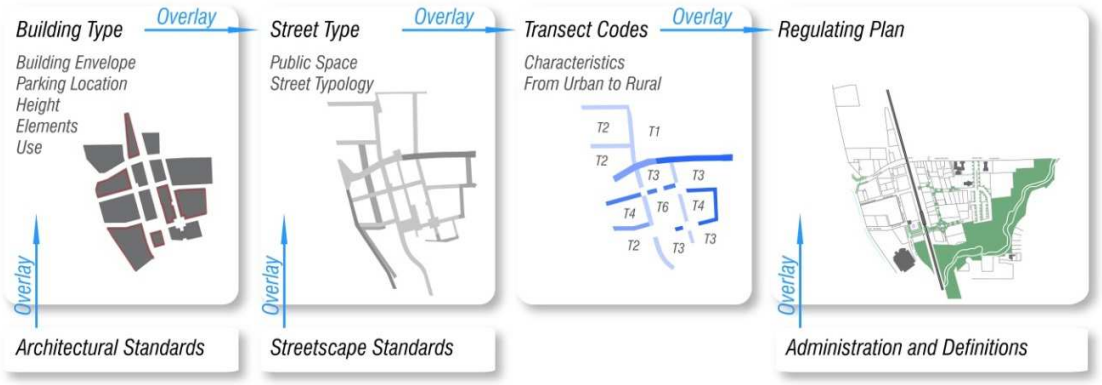


Figure 3. An overlay processes of the recent zoning codes. Adapted from Kim and Clayton, 2010.

When the building floor area increases, larger parking structure area would be required. As the footprint of the parking structure increases, the available land for the building structure decreases within the limited lot area. Considering other constraints such as building disposition, height, and density, such interrelationships among constraints make the overlay process complicate. Furthermore, when we apply given code standards to the parcel having atypical morphology, the parcel geometry affects other constraints such as front façade locations, building footprint dimensions, and building volumes (Kim and Clayton, 2010). In sum, the complex interrelationship among components, constraints, and parcel topology can delay changes and analysis not only as planners make decisions regarding the urban regulation provisions, but also as designers perform the overlay process of the code components (Kim et al., 2011).

2.3.2.3 Unpredictable development performances

One major criticism on Euclidean zoning is that urban form determined by the zoning code is unpredictable (Barnett, 2011), which caused undesirable shopping strips along the highway and residential neighborhoods separated from surroundings (Ben-Joseph, 2005). Although the recent zoning regulations place higher priority controlling urban form, a prediction of development performances is still difficult in the regulation making process (Imrie et al., 2009).

Urban form is one of major aspects of development performances of the zoning codes (Talen, 2012), relating to density and development capacity that are significant indicators for predicting environmental impacts on carbon emission, water conservation, and energy consumptions (Fischer et al., 2009; Lang, 1994; Punter, 1997). Recent zoning codes control urban form and density in diverse ways. As a density control, the number of dwelling units per acre (DU) is often used, but FAR is less used for residential buildings. However, DU is directly telling neither the building form nor density. For those in the private sectors like property owners and developers, FAR is a significant indicator to estimate the development capacities. In zoning codes, many provisions controlling urban form, such as the number of floors, maximum or minimum building width, and setbacks, affect density; therefore, understanding the relationships among regulation constraints, urban form, and density would be a fundamental step to predict development performances of the zoning codes.

2.3.2.4 Conflicts resulting from the prescriptive regulations

Shaping urban form with zoning codes increased the complexity in urban design. Talen (2009) argued that the approach governing urban form with regulations has a long history, but today's codes are more complex and difficult to implement than traditional codes. Ben-Joseph (2005, 2009) explained that attempts to reshape developments with land-use and zoning regulations can conflict with other engineering standards. For example, some building siting information can be obtained from various regulations, including planning regulations such as land-use and zoning, civil engineering regulations such as subdivision standard, architectural regulations such as building design codes, etc. In the administration section of the zoning codes, it is stated that a zoning code user shall check any conflicts with other regulations.

To avoid conflicts with other standards, planning authorities used regulation templates. The

National Building Code and the International Zoning Standards are examples of proposed universal codes (Punter, 1997). After the American Insurance Association created the National Building Code in the 20th century, the code was widely adopted. Many local agencies could use many technical standards without expense of research of individual local codes. In 1994, the International Code Council (ICC) began to develop a single set of national and international model codes for construction, sewage disposal, property maintenance, construction, and zoning. SmartCode written by DPZ is a template model for developing local zoning codes (SmartCode Central, 2011).

2.3.2.5 Inefficient data exchange

The difficult data converting process has been discussed in urban planning and urban design (Kim et al., 2010). In the urban design process, GIS provides spatial and attribute data that explain environmental and topographic information. Land-use and zoning maps can be obtained from the GIS layers. Other regulatory requirements are provided in text-based documents. In addition, the regulating plans and all figures are provided in documents. Subdivision plans created by civil engineers are generally distributed in the CAD file format. They have different scale, detail, and file formats, so they can impede the urban design process.

At a same time, the data need to be calibrated due to a wide range of data accuracy. For instance, inaccurate GIS data can delay the urban design process. Land-use and zoning information in GIS data are fundamental materials to create urban design plans. Unfortunately, they may not be directly used. Often, blocks, parcels, building footprints, and roads lines are overlapped each other. Sometimes a building footprint is outside of a property line or a parcel line is not aligned with a block line. The degree of inaccuracy of GIS data varies which would increase the complexity in the urban design process.

2.4 Computer-based Urban Design

Computer-based urban modeling enables urban design stakeholders to describe existing urban and future developments, in large part replacing the conventional use of drawings and physical models (Brail, 2008). In urban design, the use of digital models and tools has been proven to result better decisions in developing regulations and plans (Al-Douri, 2006).

2.4.1 The status of the urban design tools

Many planning authorities in the United States are using GIS to manage, monitor, and predict their communities and potential resources. GIS holds significant sources in urban planning and design such as environmental, geographic, demographic, network, structure, and aerial photo data. Land use and zoning maps are also included in GIS (Brail, 2008). Many GIS applications, however, have limited features for the urban design process (Jorge, Jose, Nuno and Jose, 2010). To provide more features for utilizing the GIS data, many software companies created urban design applications including Autodesk Civil (Autodesk, Inc., 2013), InfraWorks (Autodesk, Inc., 2014), Bentley Map (Bentley Systems Inc., 2014), CityCad (Holistic City Limited, 2014), and CityEngine (ESRI, 2014).

CityCAD was developed by Holistic City, which performs various design analyses (Holistic City Limited, 2014). Recently, various building geometries such as L-shape or U-shape masses were supported and the shading study capability was added. The tool provides a user interface and built-in object libraries for building façade and street design. The library objects support parametric modeling, but limited predefined parameters are editable such as the site information and street typologies. Additionally, the tool does not support customization by using API (Gil, Beirao, Montenegro, and Duarte, 2010).

CityEngine was initially developed by ETH Zurich, commercialized by Procedural in

Switzerland, and then acquired by ESRI in 2011. The tool is a 3D modeling software for the urban scale project, focusing on the flexible design and realistic visualization. The CityEngine also provides limited features for object customizations via user-definable application.

InfraWorks is a 3D modeling tool to create infrastructure planning and design. Formerly, the Project Galileo was developed as a test version and then commercialized as the Infrastructure Modeler. Later, it is named as InfraWorks. The tool supports layout of the urban scale projects with civil, geospatial, and building data, which can generate design proposals for civil engineers and planners. It performs various analyses such as slope, project safety, shadow study, energy consumption. It can generate rendering images and movies also. The tool can import and export files for ArcGIS, Autodesk Civil 3D, Revit, Vasari, etc. InfraWorks has limited capabilities of both object customization through user definable applications for urban and architectural design.

These tools provide capabilities for robust visualization (Gil et al., 2010). However, they have limited capabilities for urban regulation modeling. Furthermore, they provide limited features for parametric modeling and object customization through user definable applications.

2.4.2 Parametric urban design

Innumerable attempts have been made to utilize parametric modeling capabilities in urban design, but most of them focus on fast urban model creation.

Parametric Urbanism takes the idea of using parametric modeling techniques in the urban design domain (Schumacher, 2008). After Zaha Hadid and Patrik Schumacher named their design strategy in the Thames Gateway project as Parametric Urbanism, two architects have tested a series of urban digital design strategies. They demonstrated parameter manipulation in the Kartal-Pendik Master plan, and presented multiple combinations of the diverse development

scenarios. They explained that a parametric modeling approach allows urban designers to condense complex data into solutions that exhibit highly differentiated urban patterns (Schumacher, 2008).

SOM proposed “Parametric Urban Design” as a theoretical framework and created Blackbox Studio as a software prototype. Blackbox Studio is a parametric modeling platform supporting urban model creation. The software enables creating urban blocks and parcels from the user defined street grid. Then buildings can be designed by users and several parameters can be applied to the building objects. After completing the modeling process, user interfaces can present model information such as FAR or building use.

2.5 The Potential of Parametric BIM for Urban Regulation Modeling

2.5.1 BIM capabilities in the AECO industry

BIM has been developed to support the AECO industry. BIM ties all the building components with imbedded information to create a building product model (Eastman et al., 2008; Sacks et al., 2004). BIM facilitates communication of design intentions, and coordinates construction processes with the shared database. In addition, parametric design capability in BIM can enable real-time design changes as users manipulate the model data (Lee, Sacks, and Eastman, 2006).

BIM software environments couple 3D modeling with parametric form generation and rich semantics, which can express design concepts through real-time design changes. In addition, BIM platforms provide an API with OOP, which can facilitate an easy access of building information (Eastman et al., 2008; Eastman, 2009; Sacks et al., 2004).

2.5.2 Parametric modeling and OOP in BIM

Parametric modeling is one of the main capabilities of BIM that has been widely accepted in

AECO industry. Objects in parametric modeling can express geometry through parameters. The relationships among parameters can be defined through formulas. As objects store information as parameters, elements of objects can be regenerated according to the parametric values (Eastman et al., 2008; Smith and Tardif, 2009). In conventional modeling, even though a designer changes an attribute of one object in the model, the software does not keep track of the rules across the entire model. In parametric modeling, a parameter can be controlled by other parameters through formulas, the complex rule or constraints can be expressed in the parameters. Changing one parameter can have implications either in other objects or in the entire model. As such, the parametric modeling approach offers a level of flexibility responsible for constant change (Kim et al., 2011).

Parametric modeling in BIM has a potential to support development and interpretation of urban regulations (Kim et al., 2011). Parametric modeling and OOP in BIM would allow planners and code creators to test a couple of development scenarios, visualize building and block geometries, and analyze the quantitative consequences of developments in the code making process. With the parametric modeling approach, objects can represent urban form that stores the matrix among regulation constraints in parameters. As designers change object parameters, the building geometry and urban morphology may be updated, the performance of each scenarios may be analyzed, and multiple scenarios may be tested.

2.6 Summary

I reviewed the foundation of urban regulations and zoning in the United States in five aspects: background of urban regulations and zoning in the United States, characteristics of the recent zoning regulations, critiques and challenges of the recent zoning regulations, computer-based urban design, and the potential of parametric BIM for urban regulation modeling.

It is clear and significant that the zoning regulations have affected the physical character of our built environments in diverse ways. Over the years, various zoning applications have been evolved upon the political, social, and economic priorities (Holm, 2006). One consensus in regulatory intervention is shaping urban form and scale with the regulations because code reformers recognized the impact of urban from and scale on the success of sustainable developments (Ben-Joseph, 2009; Punter, 1997).

Still, both development and interpretation of the zoning regulations are complex for the urban design stakeholders. It is difficult to understand form implications of the urban regulations in that formulating geometry from the textual regulations is not always straightforward. Furthermore, the impact of zoning regulations on physical form and character of our built environment is often obscured. The zoning regulations remain difficult to incorporate into designs for the urban designer and architects.

Many urban design tools have been developed under the scope of fast generation and realistic visualization of the urban model, which is not sufficient to imbed regulatory information into the urban model. In addition, they can be used not in the regulation making process but in the urban design process. Some urban design tools provide the capabilities for urban regulation modeling, but the functionalities are too limited to support this research.

In the next chapter, selected recent zoning code samples will be analyzed to establish the outline of PURM development based on the structure, approach controlling urban form and scale, and the interrelationships imbedded in the code components and their provisions.

3 ANALYSIS OF RECENT ZONING CODE SAMPLES

In this chapter, I will review the recent zoning codes that outline urban form and scale. I will explain zoning code selection criteria and then analyze (1) the structure of zoning code components, (2) the approach controlling urban form and scale, and (3) the interrelationships across the code components and the provisions in controlling urban scale and form.

The findings will be a significant foundation to establish (1) a list of zoning code objects I will build in PUDM, (2) a set of parameters that will be imbedded in the PUDOs, and (3) an object hierarchy among the BIM objects and their parameters in the next two chapters.

3.1 Zoning Code Selection

This chapter focuses on the zoning codes that prescribe urban form and scale.

The definition of zoning codes is broad in that diverse zoning regulations have been evolved in the United States. FBC has been widely applied as many municipalities have reformed their codes based on FBC. It makes varying formats and styles of FBC. In addition, FBC has its siblings such as Form-based Codes proposed by FBCI and SmartCode. It makes the definition of FBC obscure. Therefore, it is significant to clarify the scope of the zoning code selection in this research.

One of largest selections of the recent form-based codes is presented in the Codes Study (Borys and Talen, 2013). In this research, 494 zoning codes were analyzed. 480 codes meet criteria defined by FBCI. The range of focusing area was broad from 100 people populations to 35 acres of focusing area. They classified the codes as five types: Smart Code Adopted, Smart Code in Process, Transect FBC, Other FBC, SC & FBC Discussion, and FB Guides. The majority of them are Smart Code, Transect FBC, and Other FBC. This classification is related to the utilized

zoning code template or the referenced examples. When a transect classification system is used, the code is often classified as Transect FBC. When the SmartCode template is used, it is grouped as Smart Code. Such code types may be related to the specific organization such as FBCI or code product name such as SmartCode of DPZ, so this classification is not applied in this research.

Sometimes, code titles directly explain the code type. The zoning code is often named according to the referred zoning code types or the utilized zoning code templates. Often FBC and SmartCode are used for the code title such as Station Area Form-Based Code (Farmers Branch, TX.) and Fitchburg SmartCode (Fitchburg, WI.). Sometimes, however, the title is not related to the code type as shown in Development Standards and Guidelines (Fort Worth, TX.) and Downtown Specific Plan (Ventura, CA.). Such classification using the code name is not applied in this research.

As defined the previous chapter, zoning codes and FBC are defined in this research as follows:

- The zoning code in this research refers the codes that regulate urban form and scale of open spaces, streetscapes, buildings, and parking spaces through design standards or guidelines.
- Form-based zoning codes and their various spellings such as Form-based code(s), Form-Based code(s), and Form-based design code(s) are abbreviated as FBC in this research.
- FBC in this research does not refer the specific code products created by organizations such as FBCI or by private code creators such as DPZ.
- Whether are labeled FBC or not, zoning codes in this research may have similar strategies, components, and constraints with FBC.

The regulations controlling building disposition such as Required Building Line (RBL) or Built-To Line (BTL) are widely used in the recent zoning codes. When mandatory building disposition is applied to the irregular shaped lots, the zoning code application may be complex. The zoning codes having such regulations are selected. In addition, the zoning codes having a clear project boundary will be reviewed.

In short, the criteria of zoning code selection are as follows:

- The zoning code should provide a district map that limits the project boundary. City, county, or state scale zoning codes are excluded.
- The zoning code has higher priority on controlling urban form and scale with prescriptive provisions such as RBL, BTL, setbacks, heights, and building dispositions.
- The zoning code should utilize transect classification systems for diverse scale, form, and architectural characteristics.

Based upon the selection criteria, four zoning code samples are reviewed (Table 1). They provide district maps indicating clear focusing area. The focusing area is neighborhood scale within a city jurisdiction. Standards or guidelines in these codes control form and scale of open spaces, buildings, parking structures, etc. They have governed neighborhood developments and been updated after the adoption. The type of the zoning codes in the table is based on the Codes Study (Borys and Talen, 2013).

3.2 The Review Criteria

Each zoning code is analyzed according to four criteria.

(1) Description: explains backgrounds of the project and the code development. A diagram superimposes the focusing area and the city jurisdiction.

Table 1. Zoning code samples

Title	Location	Type	Adopted year (Amended)	Focusing area (acres)	City jurisdiction area (acres)
Heart of Peoria Land Development Code	Peoria, IL	Other FBC	2007 (2011)	7,500	29,800
Near Southside Development Standards and Guidelines	Fort Worth, TX	Other FBC	2007 (2013)	1,400	223,400
Downtown Specific Plan	Ventura, CA	SmartCode	2007 (2012)	514	20,540
Form-based Code Station Area	Farmers Branch, TX	Other FBC	2005 (2012)	140	7,680

(2) Zoning code components: analyzes the major code components controlling urban form and scale, which are chapters, sections, or provisions in the code documents. I will investigate which code components will be modeled as the BIM code objects in the following chapters.

(3) Interrelationships across the code components and provisions: reveals interrelationships across code components by using specific examples. It is a significant process to formulate a hierarchy among the code components.

(4) Controlling scale and form: identifies which aspects of built environments are controlled by such code objects. It is expected to formulate a set of parameters that will be imbedded in the code objects in the following chapters.

3.3 Code 1. Heart of Peoria Form Districts

3.3.1 Description

Peoria is located on the Illinois River in Peoria County, Illinois. The Heart of Peoria is a special project district in Peoria including the downtown core and surrounding neighborhoods. The

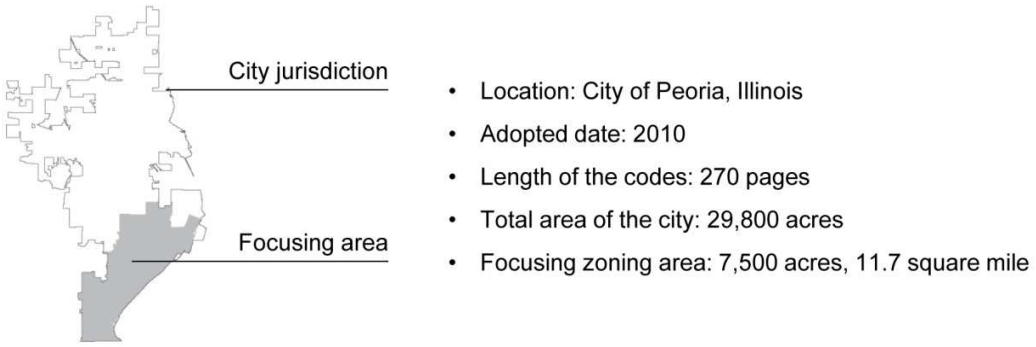


Figure 4. A zoning code description of Heart of Peoria Form Districts

Heart of Peoria plan was created to manage all developments in the special district. A transect classification system is used (City of Peoria, 2012). The code designates three special districts: Base, Form, and Overlay. No regulating plans are provided for both the Base and the Overlay districts, so it is required to use the City of Peoria Zoning District Map instead. In the other hand, the Form districts are designated along the major corridor, which shall be governed by the regulating plans and design standards. The code structure and applications of the Form districts meet the zoning code selection criteria, so the codes for the Form districts are reviewed.

3.3.2 Zoning code components

The section structure is presented in Figure 5. The colored sections show the zoning codes for the Form districts, which are reviewed. Some non-colored sections contain general rules related to the Form districts, but major provisions controlling urban scale and form are included in the colored sections. The structure of the whole zoning codes is as follows;

I review the “Form Districts” section in Figure 5, identify the code components and their structure, and create a diagram in Figure 6. In Form Districts, 4 regulating plans, 10 building envelope standards, and 7 street-type specifications are included.

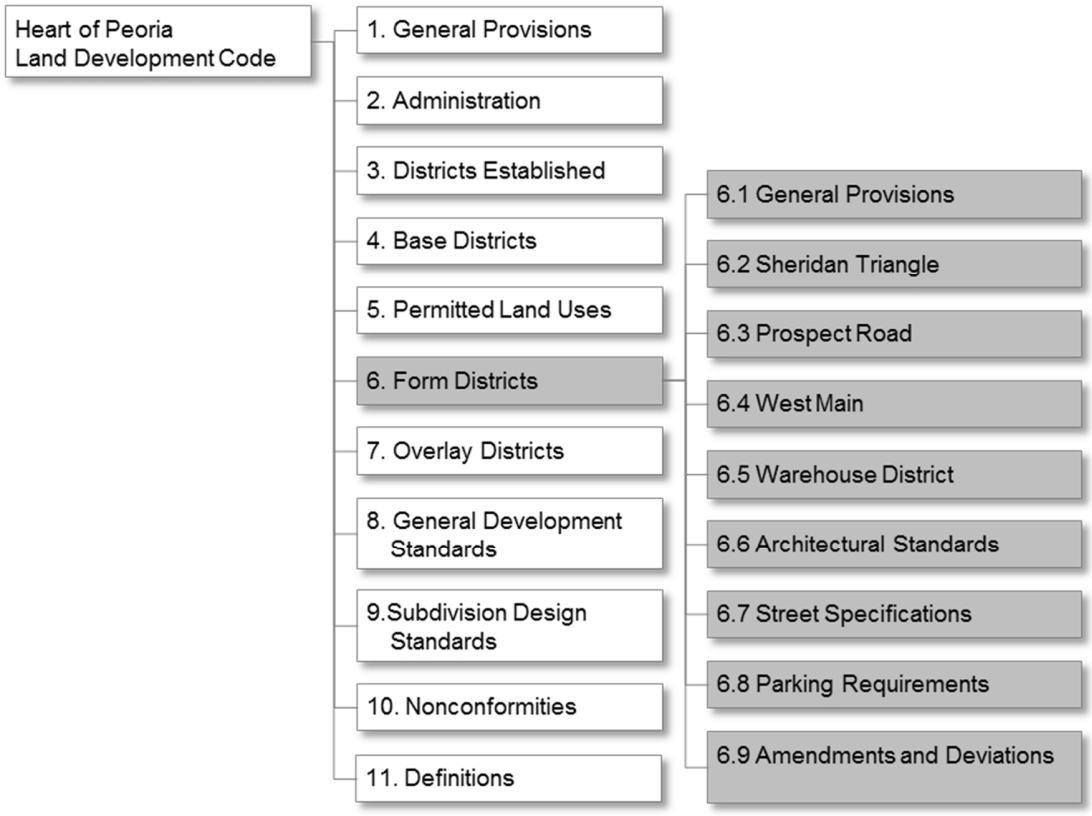


Figure 5. The section structure of the Heart of Peoria Land Development Code. Note: The grey color sections are reviewed in this research.

The first row represents the regulating plan. The second row corresponds to design standards and specifications that are chapters in the code documents. The third row shows sections under the standards and the specifications. For instance, the first component in the third row is Height represents one of four sections of the Building Envelope Standards.

The diagram consists of code components and connections. A text box in the diagram is a code component, representing a chapter or a section of the code document. A line between text boxes is a connection that visualizes the relationship among the chapters and the sections.

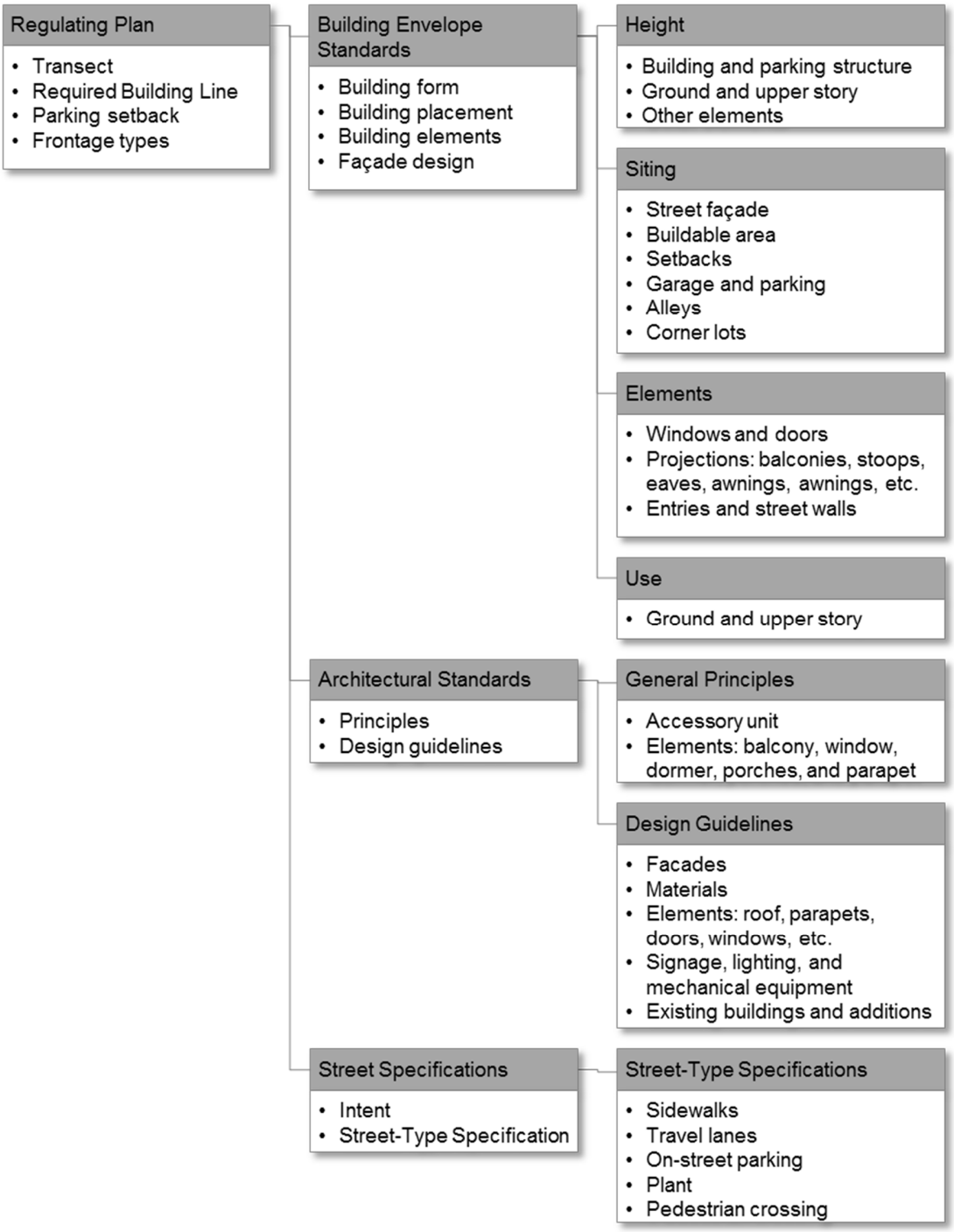


Figure 6. The code structure of Heart of Peoria Land Development Code. Note: The codes for the Form districts are illustrated.

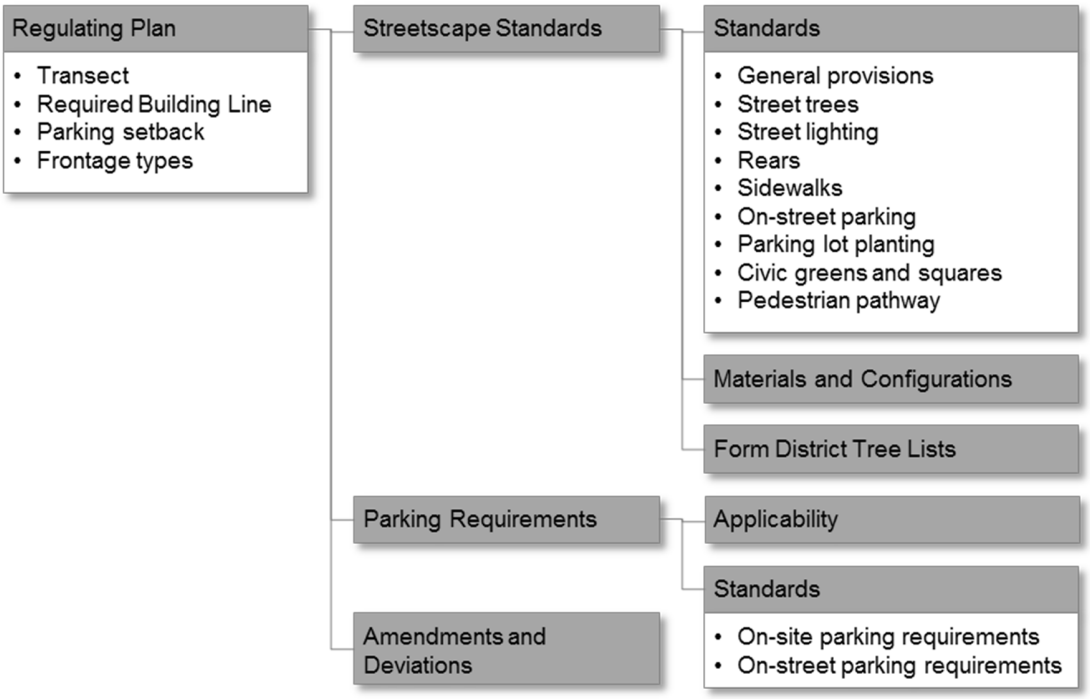


Figure 6. Continued.

The significant and noticeable relationships are modeled in the diagram. Some relationships that are not prominently related to urban form and scale are excluded. Brief descriptions of the code components are as follows:

- Regulating plans identify the disposition of each property and illustrate how each lot is related with the adjacent properties and streets. Four regulating plans are provided for four types of the Form districts. Each regulating plan defines the building frontage type, the streetscape type, the building and the parking placements, etc. The color codes are used in the regulating plans to indicate the transect codes that determine the type of the building envelope, the architectural design, and the parking standards.
- Building envelope standards govern building disposition, heights, building elements, and

use (p. 6-2). A series of the building envelope standards are provided in each regulating plan. The type of the building envelope standards can be identified from the color codes in the regulating plans.

- Architectural standards (p. 6-29) consist of design guidelines to accomplish a certain architectural character by controlling façade configurations and building components such as exterior walls, roofs and parapets, doors and windows, etc. No graphics are provided and textual provisions are provided, but no graphics are included in this section.
- Street specifications provide requirements of street configurations with traffic lanes, on-street parking spaces, planting spaces, curbs, and sidewalks (p. 6-39). For each type, a section and a plan drawing are provided.
- Street standards include general rules for public amenity design such as street trees, benches, signs, and street lights (p. 6-47). Textual provisions without graphics are provided.
- Parking Requirements govern on-site and on-street parking (p. 6-51). The required parking spaces are determined by the building use, the number of residential units, or the building footprint area. The parking setback is defined in the regulating plans, but other requirements are included in this section.

3.3.3 Approaches to control urban form and scale

This code controls urban form and scale by using provisions and guiding principles of building envelope standards and street-type specifications. This code governs block scale, building scale, building façade size, and density.

- The super scale building is prohibited. The maximum footprint area of buildings shall be

50,000 square feet (p. 6-4).

- The building frontage length is controlled. Building frontage length shall be less than 75 feet (p. 6-4).
- The density of the residential building is controlled. For residential use, maximum units per acre, minimum lot area and width, dimensions of yards, as well as heights of principal and accessory structure are defined (p. 4-2).
- A large scale block shall provide public accesses from and to surrounding area. The block face greater than 400 feet shall have accesses from or to neighborhoods such as alleys or drive ways (p. 6-3); individual properties having frontage over 250 feet shall provide such accesses (p. 6-4); and individual properties having frontage shorter than 99 feet are exempt from the requirements (p. 6-4).

Building envelope standards govern height, disposition, building element design and use. The building height is controlled as follows:

- The building height is measured in stories and the height of each story is measured in feet.
- The first floor level from the ground is measured in inches (p. 6-18).
- The height of building elements such as a mezzanine, an attic, and a street wall are measured in feet.

Building disposition is regulated by the regulating plans and the siting section in the building envelope standards. The regulating plans provide RBL and the parking setback. The siting section provides the requirements for buildable area, open space location, parking areas, etc.

- Setbacks for building and parking are defined in the regulating plan.

- A setback from the property line is measured in feet.
- The minimum area of open space is calculated in proportion to the lot area.
- The available location of the garage entries and driveways is designated in the siting section.

Some specific rules for building element design are included in the building envelope standards.

- To avoid large opaque wall design with no elements, the area of an opaque wall is regulated. In contrast, the minimum portion of the transparent area to the overall façade area is defined.
- The horizontal and the vertical dispositions of building elements, such as awnings, overhanging eaves, balconies, bay windows, and stoops, are also controlled. The dimensions are measured in feet.

The building use is controlled by the use section of the building envelop standards. The type of building use is implicitly related to the building typology, so the building use requirements may affect urban form and scale. For instance, commercial buildings generally have a higher ceiling than residential buildings.

- The use of the ground and upper story for mixed use is defined in an exclusive manner: only commercial and residential use in the ground story; no commercial use in the upper stories; and no commercial use in the accessory building.

Lastly, some standards and requirements control both configurations of streetscape and parking structures.

3.3.4 Interrelationships across the code components and the provisions

The boxes in the diagram represent zoning code components and the line connections represent the relationships between the zoning code components. For instance, a regulating plan outlines the building envelope standards. Then, the building envelope standards determine height, siting, element, and use of buildings. However, there are interrelationships across the components, which are not presented in this diagram. The following two provisions show the interrelationships that are not shown in the figure.

- "For each block face, buildings along the required building line shall present a complete and discrete vertical façade composition (i.e., a new façade design) at an average street frontage length of no greater than 60 feet for neighborhood center sites (p. 6-3)."

This provision designates the site that shall comply with the building disposition regulation according to the length of street frontage. The street frontage is defined as either a lot line or a building line coincident with the RBL (11-19). This provision imply that when a site has RBL and the lot line on RBL is no greater than 60 feet, front façade design shall comply with façade design requirements.

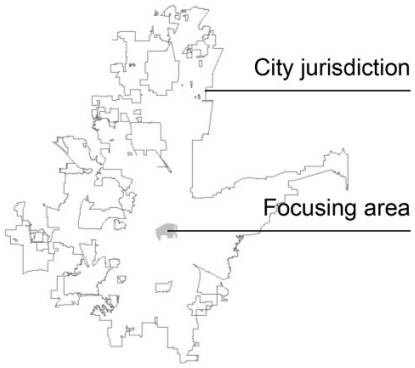
- "On each lot the building façade shall be built parallel to the required building line for at least 80% of the required building line (RBL) length (p. 6-8)."

This is a provision of Siting section under the Building Envelope Standards. This provision limits the minimum length of the building façade on RBL. With the first provision, this regulation designates the range of the building façade length on RBL. In sum, it can be inferred that two provisions make a relationship among site topology, RBL, building disposition, and the front façade dimension on RBL. To design a building front façade, such requirements shall be

satisfied, which may need a complex decision making process.

3.4 Code 2. Near Southside Development Standards and Guidelines

3.4.1 Description



- Location: Fort Worth, Texas
- Adopted date: 2007
- Length of the codes: 52 pages
- Total area of the city: 223,400 acres
- Focusing zoning area: 1,400 acres, 1.46 square mile

Figure 7. A zoning code description of Near Southside Development Standards and Guidelines

This code was adopted for the special district, Near Southside, located in the south of Downtown Fort Worth, Texas. The Near Southside is one of the urban design districts in the city of Fort Worth. The city has rigorously adopted a series of design districts, design standards, and guidelines by incorporating with non-profit organizations. The Fort Worth South Inc. (FWSI) has worked with regional stakeholders to create this code (City of Fort Worth, 2014). The code focuses neighborhood revitalization with pedestrian-oriented, well-designed, and sustainable developments (City of Fort Worth, 2008).

The code controls urban form and scale by using two transect types and three zone types. The transect classifications are “general urban” and “urban center.” Three zone types are “neighborhoods”, “institutional/industrial”, and “restricted.” Each lot shall follow standards for both the transect classification and the zone classification.

3.4.2 Zoning code components

One of the major components is the transect classification system in this code. Throughout the code components, the transect classification system is used more prominently than other zoning code samples. The main code components are presented in Figure 8.

- The transect classification (p.12) provides a continuous cross-section of built environments that this code promotes. The general urban zone (T4) and the urban center zone (T5) are mostly applied in the regulating plans.
- The regulating plan identifies the allowance of development form, scale, and use (p.15). Four districts and four regulating plans are included (p. 15). The type of transects and building use in the regulating plan are used to locate the required development standards and guidelines.
- Setback, building height, and parking disposition are defined in the regulating plan by using the transect zones. The detailed requirements can be found in the development standards and guidelines.
- The regulating plan designates the type of street specifications according to the street context and the street capacity. Three street contexts are mixed-use, main, and commercial and three street capacities are local, collector, and arterial. The detailed requirements can be found in the development standards and guidelines.
- Development standards and guidelines govern six categories of (1) streets and public space, (2) building location and orientation, (3) building height, (4) parking and drive ways, (5) architectural standards, as well as (6) on-site landscape.

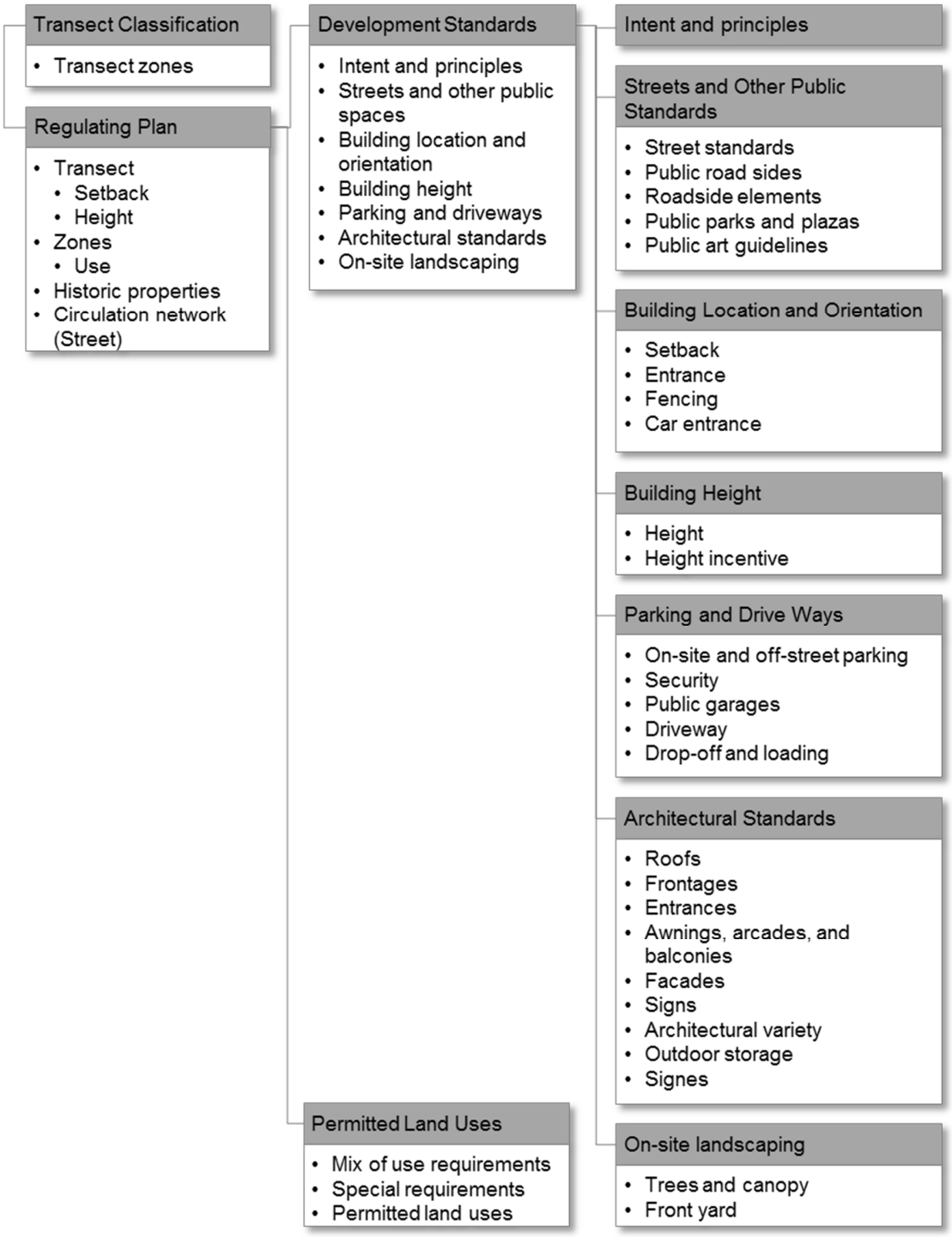


Figure 8. The code structure of Near Southside Development Standards and Guidelines

- Permitted land uses lists requirements of mixed use developments as well as the allowed or prohibited uses. According to the type of uses in the regulating plan, detailed requirements can be found in this section.

To use this code, the first step is identifying the transect classifications from the regulating plan. They are placed in the first row in the diagram. Next, the requirements can be found from the development standards and the permitted land uses. They are placed in the second row. The sections under the development standards are located in the third row.

3.4.3 Approaches to control urban form and scale

The regulating plans use a transect classification by combining two transect types and three use types. Two transect zones are T4 (general urban) and T5 (urban center). Three uses are N (neighborhood), I (institutional/industrial), and R (restricted). For example, the district in the T4-N zone needs to follow requirements for both the neighborhood use and the general urban zone at a same time (P.15). Upon the transect classification, detailed requirements can be found in the development standards.

- Street configurations and dimensions are defined the “streets and other public spaces” section and the “parking and drive ways” section. The spacing and the dimension of street trees, furniture, and lightings are included (p 31).
- The building height requirements are defined in the regulating plan and the building height section. In the regulating plan, the height of the building facade and the building mass are included.
- In the regulating plan, the building height is defined. The building height is measured by the number of stories and the height of each story is measured in feet. For instance, the maximum building height of single use in the T4 zone is 3 stories (p. 13).

- The building disposition is controlled by the regulating plans, the building location and orientation section, the “parking height” section, etc.
- The location of parking garage is decided by the main building location; a parking structure shall be behind or beside of the main buildings.
- The area of open space is measured in square feet, and the required area is decided by the ratio of the gross floor area to the open space area.
- To receive a height incentive, a certain range of open space shall be opened to public. The public space area shall not be less than 2,500 square feet and greater than 10% of the gross floor area (p.38).

The regulating plans in this code pose more regulatory information than other three samples. Generally, a regulating plan provides types of development standards and the required information can be obtained from the other chapters in the code document. In this code, however, provisions for building setback, height, and use are provided in the regulating plans.

The regulating plan does not designate RBL and parking disposition. Instead, a wide range of setbacks is defined in the building location and the orientation section. In view of building disposition control, this code may allow more flexible building design than other zoning codes having the RBL requirements.

3.4.4 Interrelationships across the code components and the provisions

Interrelationships among the code components are found throughout chapters, sections, and provisions. They are not illustrated in Figure 8, but the following provisions exemplify the interrelationships in the building height sections (Figure 9).

- Building height incentive can be determined by the ratio of mixed uses. For instance,

when a building has at least 20% residential and 10% office, restaurant, and/or retail uses, the height incentive can be applied (p.37). The ratio of each use can be measured in square foot of the floor area.

- Minimum building height, gross floor area, and the façade configuration are correlated each other. For instance, when the gross floor area is over 4,000 square feet, more than 50% of the building façade shall be higher than 18 feet (p.37).
- In the historic preservation districts, building height can be determined by the site context. For instance, buildings adjacent to the existing house in the historical districts shall comply with the height limits (Figure 9). Within 20 feet from the shared property line, 2 stories are a maximum height. Further than 20 feet from the shared property line, a 45° line on the second floor level determines the maximum building height (p. 37).

These provisions make interrelationships among the components and the provisions, which make interpretation of the zoning codes complicated.

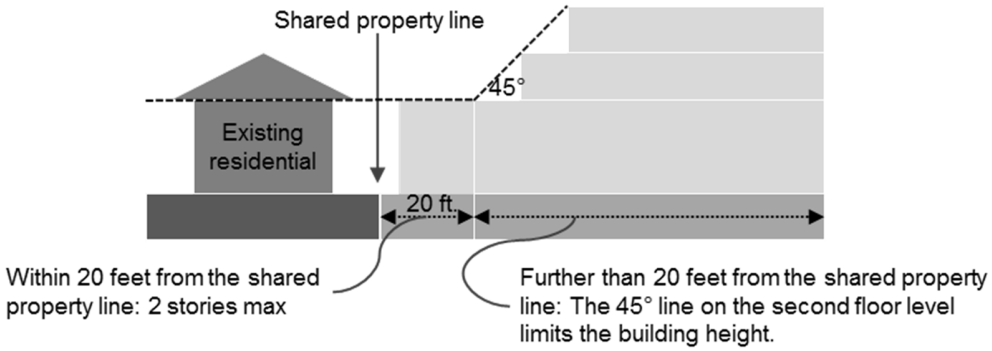


Figure 9. A building height control in the historic preservation district. Adapted from a figure, “Fairmount Transitional Height Plane” in the Development Standards and Guidelines (City of Fort Worth, 2013).

3.5 Code 3. Downtown Specific Plan

3.5.1 Description

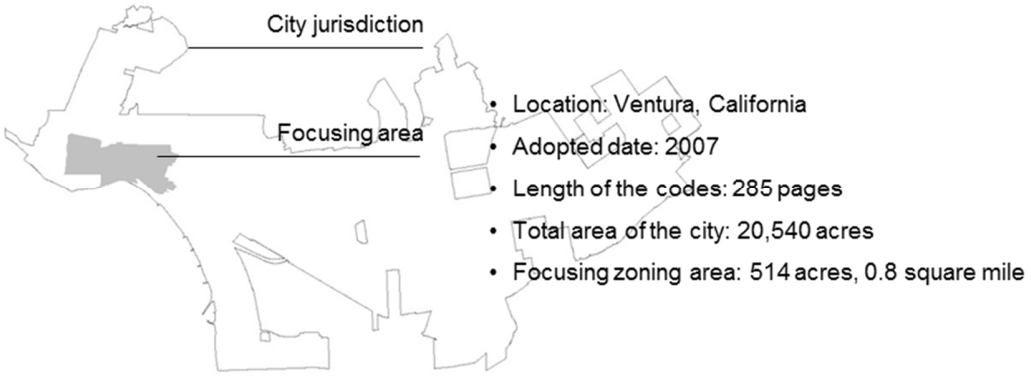


Figure 10. A zoning code description of Downtown Specific Plan

This zoning code is created for the downtown of the City of San Buenaventura (Abbreviated as Ventura in this research), California. The city is located on the California coast about 70 miles northwest of Los Angeles (City of Ventura 2007).

The first downtown plan was adopted in 1993 and it has been amended to reflect changing needs and specific goals. The major update was made in 2007 and the most recent minor revision was performed in 2012. Backgrounds, accomplishments, and the initial goals are documented in the beginning. This is the most comprehensive and longest zoning code among four samples.

In the regulating plan four catalytic projects and four focus areas are designated. For catalytic projects are multi-model transit center, cultural arts center, beach connections, and California street ramp relocation. Four focus areas are urban core, neighborhood centers, the triangle site, and beach front promenade (City of Ventura, 2007).

3.5.2 Zoning code components

This development plan consists of five chapters: Overview, Goals and Policies, Development Code, Streetscape Plan, Programs and Implementation, and Appendix. Development Code and Streetscape Plan control urban form and scale (Figure 11). I reviewed the grey color components including “Development Code”, its’ subsections, and “Streetscape Plan”. The zoning code structure within the colored components is illustrated in Figure 12. I simplify the code structure as the code objects and the connections. Descriptions of the major components are as follows:

- “Urban Standards” provides regulations for six types of transect zones. Five subsections provide the permitted land uses, the regulating plan, and specific regulations for each

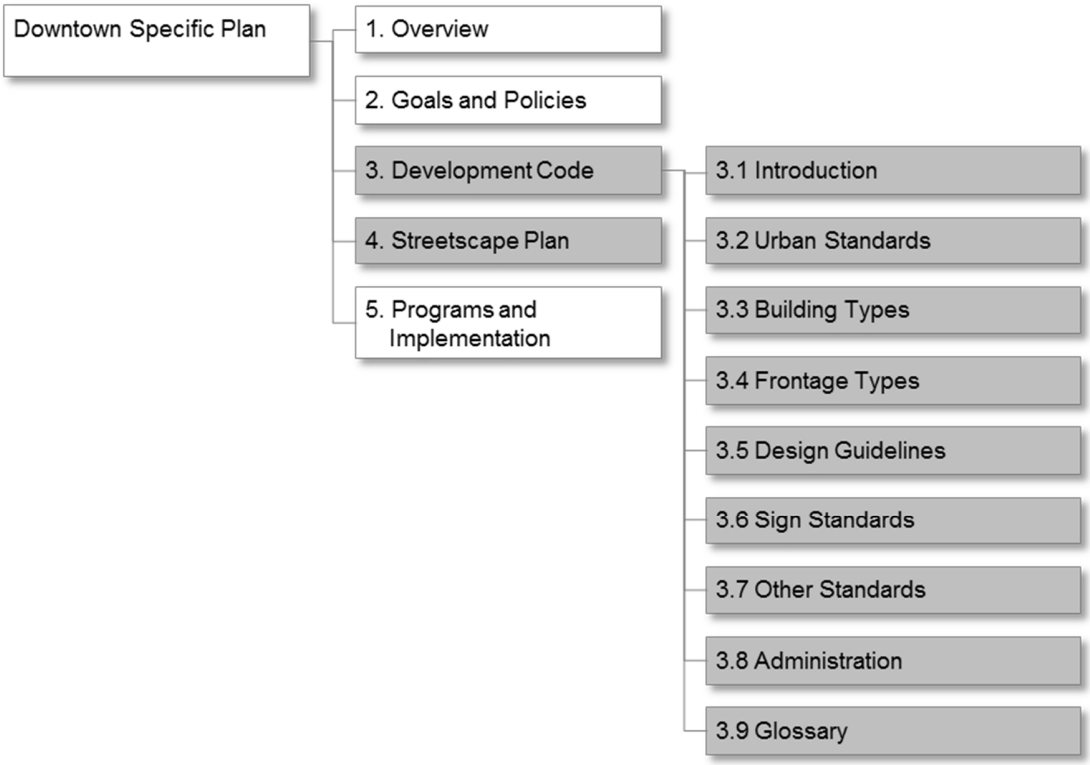


Figure 11. The section structure of the Downtown Specific Plan. Note: The grey color sections are mainly reviewed in this research.

transect zone.

- “Regulating plan” designates nine types of transect zones to assign land use and design standards. Streetscape type, RBL, or parking setback lines are not illustrated in the regulating plan.
- “Mixed Type Development Standards” in the urban standards identify the allowed location of particular uses such as commercial, bar, and nightclub. Two maps limit such development locations to the main corridor adjacent to the city hall.
- “Urban Standards by Zone” is a subsection of the urban standards. For the six types of transect zones, the requirements for the building types, horizontal and vertical configurations, and parking are defined. This section is similar to the building envelope standards of other zoning code samples.
- “Building Types” provides requirements of building types, access, parking and services, as well as open space. 13 building types are included in the code.
- “Frontage Types” identifies the allowed frontage configurations. 8 frontage types such as arcade, gallery, and shop front are defined and the type can be determined by the Building Types section. In case of the arcade frontage type, arcade depth and height, column height and spacing, as well as building façade locations near the arcade are identified (III-61).
- “Design Guidelines”, “Sign Standards”, and “Other Standards” provide recommendations for the building element design that can affect characteristics and styles of the neighborhood, streetscape, and buildings.
- “Streetscape Plan” identifies requirements and recommendations for streetscape configurations such as sidewalk, lighting, landscaping, and signage.

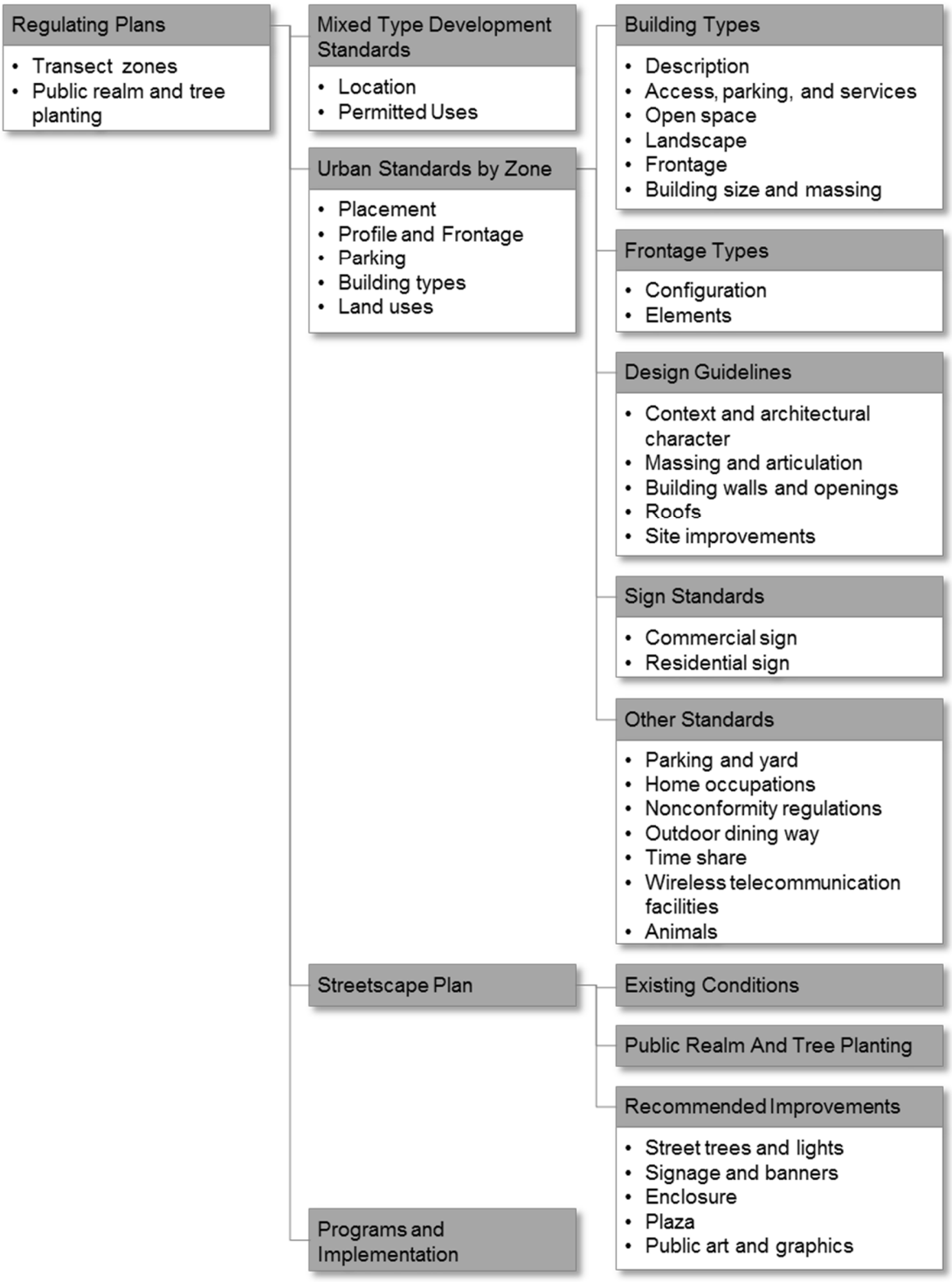


Figure 12. The code structure of Downtown Specific Plan

Following associations among components can be identified. In sum, the key component is the regulating plan.

- The regulating plan determines the transect zones.
- The regulating plan determines the type of mixed-use developments.
- The transect zone determines the type of urban standards.
- The urban standards regulate urban form and scale by controlling building disposition, height, allowed building types, frontage design, and parking structure capacity.
- The type of streetscapes is regulated by the “Public Ream and Tree Planting Plan”, which is the regulating plan for public space development.

It implies that four components are in a hierarchy according to the scale that each component governs as follows:

- The *Regulating Plan* is a guiding code object.
- The *Urban Standards by Zone* are dependent on the regulating plan.
- The Building Types are dependent on the Urban Standards by Zone.
- The Frontage Types are dependent on the Urban Standards by Zone.
- The *Streetscapes Plan* is dependent on the regulating plan.

The regulating plan is placed in the first row of the diagram. The code includes two types of regulating plans. One is a transect zone map. The other is a public realm or tree planting map. The code components dependent on the regulating plan are located in the second row. The code objects in the third row dependent on the objects in the second row.

3.5.3 Approaches to control urban form and scale

First, development density is controlled in diverse ways. To measure density of the residential

use, the number of dwelling units within a lot is counted such as units per acre (III-123). The number of occupancies of each unit is also used. The development density, the building scale (e.g., a gross floor area), and building use are determents of the required parking spaces. For instance, at least one parking space in residential use shall be provided for every 1,500 square feet floor area for the residential units in the *Urban General 1* zone (III-14).

A key component controlling urban scale and form is the *Urban Standards by Zone* section in the *Urban Standards*. According to the six types of transect zones in the regulating plan, the allowed building form and building type for the zones are defined in this component. The type of transect zone is associated to building placement, height, building types, allowed lot widths per each building type, parking, and allowed land uses are defined.

- Setback requirements affect the disposition of buildings, open spaces, and parking structures (III-14). Setbacks for the building elements such as balconies, bay windows, cantilevered rooms, and eaves are identified (III-14).
- The allowed building type and its frontage style are defined in the *Building Types* standards and the detailed requirements for the frontage configuration are included in the *Frontage Types* standards.
- The range of lot widths is determined according to the building type. The maximum width of the building frontage is regulated so that over-scale block design is prevented. For instance, the maximum lot width of *Carriage House* shall be from 25 feet to 100 feet in the *Urban General 1* zone.

The provisions controlling building form and scale are included in the *Building Types* standards. For the 13 building types, provisions control building geometry, building frontage configuration, access types, landscape, open space, and parking space. Following examples depict how building

and form are controlled in the *Building Types* standards.

- The building scale is controlled by the building types. For instance, houses and garages shall occupy no greater than thirty percent of lot area in the *Carriage House* type (III-32). The area of *Carriage House* shall be no greater than 50 percent of primary building's habitable floor area (III-33).
- The location of the lot accesses is defined. For instance, the main entrance shall be accessed from the side yard setback in the *Carriage House* type (III-32). It is related to the mass disposition.
- Some provisions control the configurations of the floor plan. For example, each dwelling unit in *Row Houses* shall have at least two facades exposed to outside light and air (III-47).

The *Frontage Types* standards govern the eight types of building frontages. Detailed requirements for the horizontal and the vertical configurations are defined (III-60). Some of them explicitly or implicitly affect both the façade configurations and the building dispositions.

- For the arcade frontage type, provisions regulate the proportion of openings to the façade, the clearstory height, the column spacing, and the bulkhead dimensions (III-60). The arcade height is associated to the first floor building height. The column spacing is related to the building façade configuration.
- In a similar way, the dimension and the location of a forecourt are defined, which can affect the building façade configuration. For instance, the minimum size of a forecourt shall be 10 feet deep and 30 feet deep. The shop front facing the forecourt shall be between 10 feet and 16 feet tall (III-62).

Some provisions prevent cake-cut building design and the monotonous façade design (III-55). Varying height ratios are asked to be applied to façade design as follows.

- In the *Commercial Block* type, only one story buildings can apply a same building height to the entire building. Otherwise, the building height shall have variations. In the three story buildings, 85 percent of a façade can be three stories. 15 percent can be four stories. In the four story buildings, 75 percent of a façade can be four stories. 25 percent can be five stories.

The Downtown Specific Plan has been amended since the first plan was adopted in 1993. Over 20 year period for code amendments makes this code more comprehensive and detailed than other three code samples. Some provisions reflect higher priority on the history preservation and some provisions control urban form and scale in a prescriptive approach.

3.5.4 Interrelationships across the code components and the provisions

There are complex relationships among code components. For instance, a type of building frontage is defined in *Building Types*. A type of building is defined in *Urban Standards*. A type of *Urban Standards* is defined in the regulating plan. The following example shows the process to design a commercial building in the urban core zone.

(1) Check a zone type: when a lot is defined as the *Urban Core* zone (T1) in the regulating plan, *Urban Standards* for *Urban Core Zone* need to be used.
(2) Check a building type: *Commercial Block* is one of the allowed building types in T1.
(3) Check provisions controlling building form: the placement, setbacks, and height requirements are defined in *Urban Standards* (III-24).
(4) Check provisions controlling the lot dimension: the allowed lot width for the commercial

block is defined in *Urban Standards* (III-24). With the setback requirements, the allowed width relates to the front façade length.

(5) Check provisions controlling building height: the height ratio requirements needs to be identified from *Building Types* for *Commercial Block* (III-54).

(6) Calculate the height ratio: the height ratio is calculated by the façade area ratio for each story. The height ratio in the *Urban Core* zone is defined as follows (III-55); "Commercial Block may exceed the 75% 4-story, 25% 5-story limitation of the T6.1 Urban Core zone."

(7) Check provisions controlling open space: when a forecourt is designed on the shop front, the *Frontage Types* for forecourt section (III-62) needs to be complied. The dimensions such as width and depth are identified.

(8) Check provisions controlling building facade: to design a front façade, the all requirements above need to be synthesized.

In this process, the regulating plan, the urban standard types, the building types, and the frontage types should be used at a same time, but they are explicitly or implicitly related to each other. One interrelationship imbedded in the example is discovered in the height ratio requirements (6). The provisions for the height ratio requirements make an association among the façade area and the building footprint area per building use, and the building height. It is also can be inferred that the building footprint area is related to the provisions such as placement, setbacks, and height in *Urban Standards* (3).

3.6 Code 4. Form-based Code Station Area

3.6.1 Description

Farmers Branch is a city of the Dallas-Fort Worth metropolis, which is located north of Dallas.

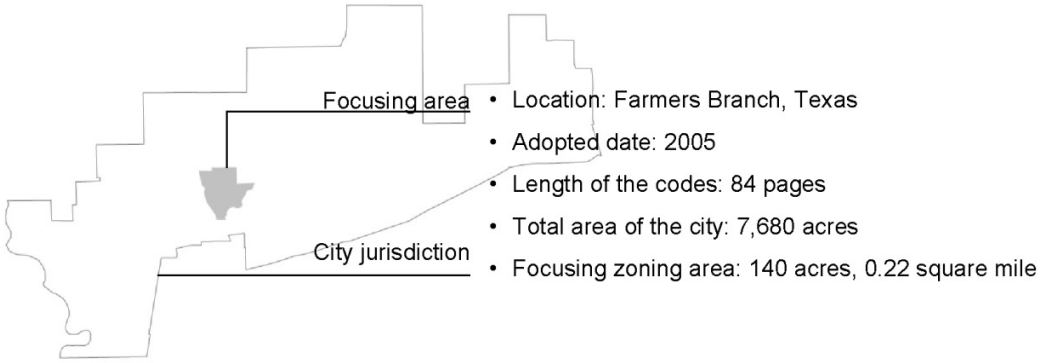


Figure 13. A zoning code description of Form-based Code station area

This code was prepared for developments near the transit station as a part of the Dallas Area Rapid Transit (DART) extension plan. To establish safe and attractive environments near the DART station, the city created the code, "Station Area Form-based Code" and the master plan, "Farmers Branch Station Area Conceptual Master Plan." Since the master plan was adopted in 2002 and this code was adopted in 2005, they have been amended through public meeting and workshops (City of Farmers Branch, 2014).

The code focuses on controlling coherent physical form and design styles as well as flexible building use and density (City of Farmers Branch, 2005). The code provides two types of regulating plans for the flexible application.

This code prevails over the zoning ordinance when a conflict is found. The portion not covered by this code has to comply with the zoning ordinance (City of Farmers Branch, 2005).

3.6.2 Zoning code components

The major components of this code are illustrated in Figure 14. The style and the format of zoning code components are not significantly distinctive to other three samples.

- Three types of plans are included: Regulating Plan, Urban Design Plan, and Regulating Plan Street Types. *Regulating Plan* identifies a project boundary, the building frontage types, street designation, RBL, and the parking setbacks. *Urban Design Plan* identifies five types of urban centers, street tree alignment, and building height per blocks. *Regulating Plan Street Types* identifies the location of five street types.
- *Building Envelope Standards* govern three dimensional building placement and the building elements such as shop fronts, windows, balconies, etc. Six types of the building frontage are identified in the regulating plan and governing provisions are included in *Building Envelope Standards.*
- The section of *Street Types* is a sub section of Regulating Plan, which specifies typical configurations for vehicle lanes, sidewalks, and public amenities. Five street types are designated in the regulating plan. The special requirements for each street type are specified in *Streetscape Standards*.
- *Streetscape Standards* provide textual provisions controlling materials and configurations of streetscape that are not defined in *Street Types*. The standards cover minimum standards for trees, sidewalks, and on-street parking, materials and configurations of open spaces, and the type of trees in the station area.
- *Architectural Standards* governs coherent architectural styles through materials, configurations, and construction types. The scope of *Architectural Standards* is the building elements such as exterior walls, roofs, doors, and windows.
- The *Administration* section describes intention and enforcement of the provisions.

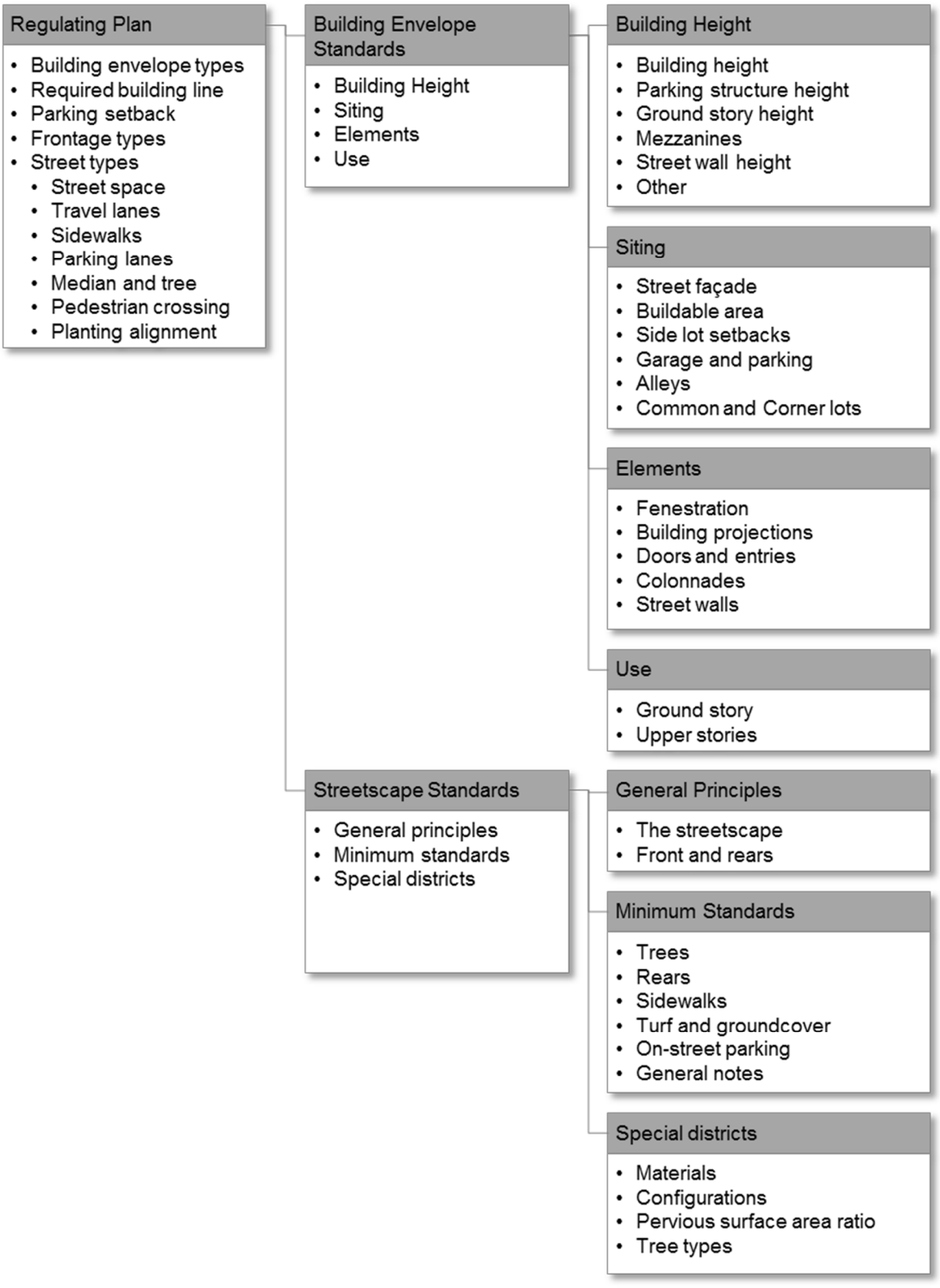


Figure 14. The code structure of Form-based Code Station Area

Figure 14. Continued

As I discussed previously, there are relationships among zoning code components, which are apparent to define. I simplify the chapters and the sections and create the abstracted code

components. The code components are arrayed in the three rows upon following rules.

- The code components in the first row represents the regulating plan.
- The code components in the second row represent three standards.

The code components in the third row represent the sections under the standards. For instance, *Building Envelope Standards* has four sections; height, siting, elements, and use. They are placed in the third row under *Building Envelope Standards*.

3.6.3 Approaches to control urban form and scale

In this code, some provisions shall be applied to entire districts regardless of the transect classifications. There provisions are named as general rules. In contrast, some provisions shall be applied to specific building frontage types or the particular street types. They are called as specific rules in the code document. Provisions controlling form and scale are discovered from both of them.

Some general rules control development scale to prevent superblock development.

- Over scale building is not allowed. The area of the building footprint is no greater than 60,000 square feet (p.15).
- Long linear building façade is not allowed. An average building length of a street frontage is no greater than sixty feet (p.15).
- Large scale projects have to provide appropriate accesses such as alleys, drive ways, or pedestrian pathways. The individual lot having over 250 feet frontage lines has to meet the requirements within the lot, unless the block in which the lot is located already satisfied the requirements within the block (p.15). On the other way, a small lot having frontage lines less than 75 feet are exempt from the requirements (p.14). Without an

appropriate access, the length of the block faces is no greater than 400 feet (p. 14).

Building height is controlled by Urban Design Plan and Building Envelope Standards. When conflicting, Urban Design Plan prevails over Building Envelope Standards.

- In *Urban Design Plan*, minimum building height is designated along important corridor. "Minimum (building) height along Buttonwood Street" is three stories (p. 21).
- The *Height* section of Building Envelope Standards designates minimum and maximum building height, ground story height, upper story height, and street wall height. It is mentioned that the provisions shall be used with Urban Design Plan.

Some general rules control parking spaces and streetscapes, which can directly or indirectly constraint the development scale.

- The minimum parking requirements are defined according to the lot area and building use. For instance, a residential building in a site over 20,000 square feet shall provide more than 1 and 1/8 parking space per unit (p. 16).
- The scale of the streetscape such as street trees and street lights is regulated in *Regulating Plan*. An average interval of street lights is no greater than 75 feet and the height has to be between 9 and 16 feet from the ground level (p. 16).

One the other hand, the specific rules control urban form and scale, which are discovered in *Building Envelope Standards*, *Streetscape Standards*, and *Architectural Standards*.

- Six types of *Building Envelop Standards* provide provisions controlling building height. The types of building envelope are defined in Urban Design Plan. The length of building façade is also defined in *Building Envelope Standards*.

Building disposition is regulated by the specific rules in the *Siting* section of the *Building Envelope Standards.* The RBL requirements prescribe building disposition.

- The length of the building front façade on the RBL is controlled according to the RBL length.
- Site-specific setback requirements are defined in *Building Envelope Standards*.
- The area of the open space, uncovered or permeable surface, is measured by the percentage of the property area. Such open space shall be located behind the parking setback.

In view of code component structure, this code is not distinctive from other zoning code samples. *Regulating Plan* assigns the building frontage types and the streetscape types using the transect classification system. This code provides three types of plans including *Regulating Plan, Urban Design Plan,* and *Regulating Plan Street Types.* Each plan is easy to read, but code users need to overlap three into one.

The architectural standards regulate building façade design regardless of the type of building envelope and the streetscape. In other words, under the different building envelope types, identical architectural standards can be applied.

3.6.4 Interrelationships across code components and provisions

General relationships among the code components can be understood from the diagram. *Regulating Plan* assigns the type of building envelope and streetscape. The Building Envelope Standards control the height, disposition, the elements, and the use of buildings. It can imply that the building height is dependent on the type of the building envelope and that the building envelope type is dependent on the regulating plan. A similar relationship exists in *Streetscape*

Standards.

However, not all relationships of components and provisions are presented in the figure. Following examples show provisions pose interrelationships among code components.

- The building façade shall be positioned to the RBL or GFRBL. At a same time, the length of the building façade on RBL or GFRBL shall be at least 85 percent of the RBL or GFRBL. GFRBL means the RBL for the ground floor. The length of the RBL and the GFRBL depends on the site, so this provision makes an association among the building façade dimension, the building façade location, the RBL, and the GFRBL.
- When the *Shopfront Colonnade* site is located within 40 feet of an existing single family residential zoning district, the maximum height of eave or parapet shall be 32 feet (p. 35). This provision makes an association between the building height and the distance from the designated zone district.
- The minimum open space area is regulated in proportion to the lot area and the open space location is regulated by the setback lines. This provision makes an association among the site topology, the setback requirements, the designed building footprint, and the parking spaces, which is very similar to other zoning code samples.

3.7 Summary

I reviewed four sample zoning codes to identify (1) the structure of the zoning code components, (2) interrelationships across the zoning code components and the provisions, and (3) the approach in controlling urban form and scale.

I conclude that the components of four zoning codes have a similar structure. A wide range of provisions pose interrelationships across the components, which are neither consistent nor

obvious to understand. In addition, four zoning codes control similar elements of built environments with a similar set of provisions, while they pursues specific goals and objectives upon varying regional, cultural, and economic priorities.

3.7.1 The structure of the zoning code components

The code components of four codes can be categorized by the urban design criteria comprised with district map, zoning, public space, streetscape, building envelope, parking, and architectural design. Four codes have similar code components despite diverse naming conventions. Table 2 shows that all sample codes provide the components that correspond to district maps, the public space standards, the streetscape standards, the building envelope standards, the parking standards, and the architectural design standards.

3.7.1.1 A relationship among the code components

As shown in Table 2, all codes provide a set of district maps. The regulating plan is the key components of the zoning code samples and the following relationships are discovered among the code components.

- A regulating map determines the zone types.
- The zone type determines the type of standards for public space, streetscape, building envelope, and parking.
- The type of building envelope standards determines the type of architectural design standards.
- The type of buildings envelope standards determines the parking standards. The zone type can be the other determent.

Table 2. Major components of the zoning codes

Title	Code 1: Heart of Peoria Land Development Code	Code 2: Near Southside Development Standards and Guidelines	Code 3: Downtown Specific Plan	Code 4: Form-based Code Station Area
District map	• Regulating Plan	• Regulating Plan	• Regulating Plan	• Regulating Plan
Zoning (Transect/uses)	1	• Transect Classification	• Transect Classification	2
Public Space	• Street Standards	• On-site landscaping Standards	• Public Realm and Tree Planting Standards • Sign Standards • Other Standards	• Urban Design Plan
Streetscape	• Street Specification • Street Standards	• Streets and Other Public Standards	• Streetscape Plan	• Street-Type specificatio • Streetscape Standards
Building Envelope	• Building Envelope Standards	• Building Location and Orientation Standards • Building Height Standards	• Building Types Standards • Frontage Types Standards	• Building Envelope Standards
Parking	• Parking Requirements • Parking Standards	• Parking and Drive Ways Standards	• Parking Standards	3
Architectural Design	• Architectural Standards	• Architectural Standards	• Design Guidelines	Architectural Standards

Note: Following sections correspond to the code components.
1. Building Envelope types and Street Frontage types in the regulating plans
2. Frontage types and the street types in the regulating plans
3. General rules for parking are included in the Regulating Plan.

- The site topology in the regulating map is a determent of disposition requirements for buildings, parking structures, and open spaces in that RBL, building setback, and parking setbacks reflect the site topology.

The general relationships among the zoning code components are similar to the overlay process defined in the zoning code documents. Each code document has a user manual section explaining how to assemble the code components. In general, the first step is reading the regulating plan. Then it is asked to define the type of zones, streetscapes, buildings, etc. Other design guidelines or recommendations can be read lastly. In short, four code samples can be used with the similar overlay process.

3.7.1.2 Varying formats of the regulating plans

The amount of information in the regulating plans varies over four codes (Table 3). In the code 1, the regulating plan designates the type of frontage types, setback lines, and RBL. In the code 2, building use is described in the regulating plan. The code 3 does not describe RBL and setbacks with the regulating plan, but the code provide corresponding provisions in other code components. In addition, missing information from the regulating plan is included in other zoning code components or in other types of plans.

Table 3. Regulatory information in the regulating plan

Catetory	Code 1	Code 2	Code 3	Code 4
Building use	X	O	X	X
Transect code	X	O	O	X
Streetscape type	X	X	X	O
Building frontage type	O	O	O	O
Openspace type	X	X	X	O
RBL and setback	O	O	X	O
Parking setback	O	O	X	O

3.7.2 Similar approaches to control urban form and scale

Four codes regulate very similar criteria of built environment such as streetscape, open space,

Table 4. Major criteria in the streetscape standards

Catetory	Code 1	Code 2	Code 3	Code 4
STREET CONFIGURATION				
• Streetspace	O	O	O	O
• Sidewalks	O	O	O	O
• Street trees	O	O	O	O
• Street lights	O	O	O	O
• Median	O	O	O	O
• Travle lanes	O	O	O	O
• Pedestrian	O	O	O	O
• Pedestrian crossing	O	O	O	O
• On-street parking	O	O	O	O

building, and parking space. In the zoning code components, they use similar styles of provisions. In the streetscape standards, four codes regulate the street configuration such as travel lanes, on-street parking, pedestrian crossing, sidewalks, street trees, and street lightings (Table 4). The total number of travel lanes and the allowed dimensions of road and sidewalks are also defined.

Four codes regulate the very similar elements of built environment. In the building envelope standards, they regulate building height, disposition, and building design elements. In terms of the provision style, four samples are similar each other. For instance, the building height is controlled by the number of floors and the height of each floor is measured in feet.

However, the way of controlling building scale varies over the codes. For instance, the code 1 and 3 designate the maximum and minimum development density by using the number of residential units per site area and the number of occupancy. In the code 1 and 4, the building scale is regulated by using the maximum floor area and FAR. The scale of building envelope is also controlled by using the front façade length (the code 1 and 4) and the front façade area (the

Table 5. Major criteria in the Building Envelope standards

Catetory	Code 1	Code 2	Code 3	Code 4
BUILDING SCALE				
• Density	O	X	O	X
• The maximum floor area	O	X	X	O
• The front façade length	O	X	X	O
• The front façade area	O	O	X	X
BUILDING USE				
• Permitted uses	O	O	O	O
• Use per floor	O	O	O	O
BUILDING HEIGHT				
• Ground level	O	X	O	O
• Building height	O	O	O	O
• Parking structure height	O	O	O	O
• Ceiling height	O	O	O	O
• Accessory structure	O	X	O	X
BUILDING DISPOSITION				
• RBL	O	X	X	O
• GFRBL	X	X	X	O
• Building setback	O	O	O	O
• Parking setback	O	O	O	O
• On-site open space	O	O	O	O
• Access, alleys, and entrance	O	O	O	O
BUILDING ELEMENTS				
• Fenestration area	O	O	O	O
• Windows	O	O	O	O
• Doors	O	O	O	O
• Attic	O	X	O	O
• Awnings, balconies, and stoops	O	X	O	O
• Eaves, parapets, and street walls	O	X	O	O

code 1 and 2).

In building disposition control, RBL and GFRBL are not used by all codes, but they have provisions controlling setbacks of building and parking structures. Their setback regulations

prescribe the building façade location and orientation in that the building façade should be located on the designated setback lines.

3.7.3 Interrelationships across the code components and the provisions

As I discussed in each zoning code review, various provisions controlling urban form and scale pose interrelationships across the zoning code components and the provisions. When a provision regulates form and scale of buildings and parking structures, interrelationships are often discovered. For instance, on-site parking is associated with parcels, buildings, and open spaces. The required number of parking spaces is related to the building floor area and use. The disposition of the parking structure is related to the parking setback requirements. The available footprint area for the parking structure is associated with the area of lot, building footprint, and open space. This interrelationship is imbedded in four zoning codes. I will implement an application to demonstrate how such interrelationships in the code document can be established in BIM. A detailed procedure and the results will be discussed in the application development chapter.

In the next chapter, a set of components I formulated will be built in the PURM, components' relationships will be represented as a hierarchy of BIM objects, and a series of provisions will be parameterized in the PUDOs.

4 PARAMETERIZATION OF URBAN REGULATIONS WITH PUDM

In this chapter, I will describe development of the PUDM and its' components, the PUDOs. First, I will explain how the PUDM is constructed using the PUDOs. Then I will explain the modeling process of the individual objects in view of an object hierarchy, object parameters, and geometric configurations.

4.1 Introduction

Various urban regulations have been evolved with high priority in controlling form and scale of built environments (Barnett, 2011; Ben-Joseph, 2009). Recent urban regulations widely use prescriptive provisions controlling urban form and scale. Often one aspect of urban form and scale is determined by multiple provisions located in various zoning code components, so the recent urban regulations pose interrelationships among the regulation components and the provisions (Kim et al., 2011).

The review of recent zoning codes in Chapter 3 implied that (1) the selected zoning codes focus on the similar urban design criteria such as open space, streetscape, building, and parking, (2) they have similar code components (Figure 1), (3) they regulate similar geometrical entities of urban form, as well as (4) they present interrelationships among the components and the provisions.

Based upon the implications, I establish five types of code components (Site, Block, Parcel, Building, and Parking), a hierarchy among the code components, provisions controlling geometrical entities of urban form, and relations among provisions. Then I create PURM, PUDM, PUDOs, and PUDAs. Their definitions are as follows:

(1) PURM is a new urban planning and design platform. The model consists of PUDM and

PUDAs.

(2) PUDM is an urban district model in BIM. The model consists of a series of PUDOs. In view of the recent urban regulations, the PUDM corresponds to the regulating plan. The development process of the PUDM is explained in this chapter.

(3) PUDOs are geometrical representations of built environment that the urban regulations aim to establish. Five types of the PUDOs are created including Site, Block, Parcel, Building, and Parking. Individual objects represent the regulation components defined in Chapter 3. The development process of the PUDOs is illustrated in this chapter.

(4) PUDAs establish relations among the PUDOs and their parameters, which can represent interrelationships among the regulation components and provisions. The applications also perform complex operations that cannot be handled only by the PUDOs such as updating multiple parameters within multiple Code objects, manipulating geometries of multiple PUDOs, checking regulation compliances, and testing multiple design options. The applications are written in the BIM API and an OOP language, which will be described in Chapter 5.

In this chapter, I will describe how to create individual PUDOs using a parametric modeling approach in BIM and how to construct the PUDM with the individual PUDOs.

4.2 Methodology

4.2.1 Parametric modeling in BIM

One significant method is parametric modeling in BIM that has been widely accepted in the AECO industry. BIM ties all the building components with extensive information to create a building product model (Eastman et al., 2008; Sacks et al., 2004). BIM, on the one hand, facilitates communication throughout the design, the construction, and the facility management

processes with the shared database. Parametric modeling capabilities in the BIM authoring tools allow BIM objects to store knowledge-based rules and experts' knowledge into parameters so that BIM performs rule-based automatic manipulations and semantically-rich model generations (Sacks et al., 2004). The approach of parametric modeling is not only new but also invented by either BIM or the BIM authoring tools. However, parametric modeling capabilities in BIM provide mechanisms to store both domain-specific rules and experts' knowledge into the building components in BIM, which make parametric modeling in BIM distinctive from others.

Conventional CAD systems used coordinate-based geometry creation. Two dimensional drawing are generated from the designed object and then graphical entities were added to represent design elements such as a wall or a roof. Later surface and solid modeling features were added for more complex form generation. But the coordinate-based geometric model is difficult to modify and the drawings cannot be automatically synchronized with the updated model for each change.

Parameters either store objects' interrelationships or trigger behavior of an object. Once individual object's transitions (check) are associated with each other, a parameter change on an object can determine other objects' behavior. For instance, relocating one wall in the parametric model can enforce other related objects such as windows on the wall to be adjusted accordingly.

The capabilities of parametric modeling in the BIM authoring tools may not prevail over other parametric modeling tools. However, once a parametric model is created in BIM, various building design can be performed from the parametric model by using various modeling capabilities and the plentiful libraries of building design elements in the BIM authoring tools. For instance, a building bulk in urban design can be built with a mass object in BIM. Then the mass surfaces can be used to create building envelope and the building components in the library such as wall, roof, and floor can be placed on the building envelope.

In development, Autodesk Revit is used as a BIM authoring tool. In the Revit terms, I create PUDOs in the Revit Family level and combine them in the Revit Project level to build the PUDM. The Revit Family is an object that allows object parameters to be associated with the object geometrical and non-geometrical attributes. This capability enables the object geometry or dispositions to be assigned by the object parameters and allows various non-geometrical attributes to be stored in the object parameters. In development, geometrical transformation of the PUDOs is driven by the parameters. In addition, some parameters are built to store non-geometrical attributes within the PUDOs.

4.2.2 Behavior and attributes of a BIM object

The PUDOs and the PUDM can be created based upon domain-specific rules in the urban regulations. Existing BIM objects represent architectural domain-specific components such as walls, roofs, and floors. They have functionalities that can respond to model changes in the real-world of the architectural domain. For example, increasing wall width from the wall parameter makes the wall object longer in BIM. Increasing the roof angle parameter makes the roof steep. These functionalities are predefined behaviors of the existing BIM objects.

A wall object in BIM can be moved, expanded, or rotated when parametric definitions are updated. Such transitions are predefined behaviors of the wall object in BIM. However, default BIM objects are not appropriate for the PUDM creation in that they correspond to architectural domain specific behavior (Kim et al., 2013).

User definable objects in BIM allow a BIM object to establish custom behavior and special attributes to express various geometric representations in way of parametric modeling (Kim et al., 2013). Thus, PUDM and its objects are created with the user definable objects in BIM, enabling to establish domain specific behavior and attributes found in the urban regulations.

4.2.3 Form, function, and behavior of PUDOs

Identifying behavior of BIM objects is important, since an object in BIM can be defined with its form and behavior. In this research scope, behavior of a code object can determine actions of the object in BIM, corresponding to the urban regulation constraints. For instance, allowed building height in the zoning codes can be represented as an action that changes the height dimension of the building objects in BIM. Changing height can be identified as behavior of the building object in BIM.

The concept of form, function, and behavior is important to identify and formulate required behavior from the textual regulations. The initial concept of form and function is traced to Sullivan's article about the tall office building design (1896). In this article, he asserted that "form ever follows function", which implies that the intended purpose of a designed object is a determinant of its shape (Clayton, 1998). This notion influenced modern architecture in the 20 century.

In design cognition and artificial intelligence research, Gero's Function-Behavior-Structure (FBS) model (1990) has been expended to cover broader cognitive issues in the design process (Gero, 1995; Gero and Kannengiesser, 2004). His notion is that a designed object can be described by three variables; function, behavior, and structure. Function is the intent of the object. Behavior is the way the object performs. Structure is the object itself. Behavior can be either derived from the structure itself or specified by the requirements or expectations of the functions. The FBS ontology has been widely investigated in artificial intelligence and design cognition research.

In BIM research for product modeling, form and behavior of building products have been investigated to describe the physical characteristics of building products (Sack et al, 2004;

Ghang, 2006; Eastman, 2009). These studies focus on development of the parametric modeling platform for precast concrete buildings. Sacks defined behavior as "automatic system actions that maintain the topological and geometrical consistency to the relationships within and between model objects (Sack et al., 2004)." Ghang defined behavior as "the ability of an object to respond to internal or external stimuli." They mentioned that this definition differs from Gero's FBS model (Ghang et al., 2005) and there is no further discussion on function.

Clayton (1998) defines form, function, and behavior of objects in the virtual product model (VPM) as follows: "forms are the geometry and materials of the artifact.", "Functions are the required and desired qualities of the artifact.", and "Behaviors are the response of the artifact to particular conditions."

In this research context, I define form, function, and behavior based on and the Sullivan's notion and the Clayton's research (Clayton, 2006; Clayton, Teicholz, Kunz, and Fisher, 1999).

- Form: the geometry of the built environment resulted from the urban regulations.
- Function: the desired qualities of the built environment that the urban regulations pursue.
- Behavior: the response of the built environment to the particular urban regulations.

In this research, clarifying behavior of the PUDOs is an essential step in parameterization of the urban regulations in BIM. For instance, an open space standard in the urban regulations secures a minimum open space area for each lot. A series of provisions controls building form such as width, depth, and height, as well as building disposition with setbacks and front façade locations. Controlling building dimensions and dispositions can be understood as behavior of the BIM objects for open space standards. The allowed value such as building height and depth can be an attribute accordingly.

4.3 PUDOs in the PUDM

Within this research context, I create five types of PUDOs, assemble them, and construct PUDM.

PUDM visualizes form and scale of built environment that code components and provisions aim to establish. PUDM represents community scale development plans that have public spaces such as open spaces, traffic lanes, and pedestrians as well as private spaces such as buildings, parking structures, and private yards. In view of the recent urban regulations, PUDM is primarily based on the regulating plan, the urban design standards, the building envelope standards, parking standards, and open space standards.

The recent urban regulations provide graphics and drawings to explain how provisions regulate urban form and scale. The building envelope standards, for instance, provide the plan, the section, and the elevation drawings. Typically these drawings exemplify generic sites then the urban regulations inquire designers to apply the codes to the specific sites in the regulating plan. Often site-specific issues related to particular provisions are discovered, but they are not presented in the graphics and drawings.

In the PUDM, the PUDOs are provided for all parcels, which can visualize site specific form implication of the urban regulations. The individual objects in the PUDM can be transformed as parameter values are updated by Parametric Code Applications. In the development process, the PUDM provides the development framework of PUDAs. The object hierarchy and the type of parameters are key information in creating applications.

4.3.1 PUDOs and parameters

The PUDM consists of five types of PUDO in BIM; Site, Block, Parcel, Building, and Parking

(Figure 15).

The Site object represents the entire project boundary that includes traffic lanes, public open spaces, and developable properties (private parcels). The Block object represents pedestrian spaces in-between traffic lanes and private parcels, representing on-street parking, sidewalks, and walkable public spaces. Streetscapes such as street furniture, street trees, and landscapes are placed on the Block object.

The Parcel object is an individual parcel where building structures can be built. This object holds site topology and regulatory information that affect open spaces, building dispositions, and building envelopes. For instance, the minimum ratio of open space to the property area and the maximum area of the building footprints are controlled by many code provisions. The area of the Parcel object is a determinant factor of the minimum open space area and maximum building footprint area. The property topology is related to the building setback lines and façade location, affecting building disposition (Kim, et al., 2013).

The Building object represents a buildable building volume within a parcel. This object holds

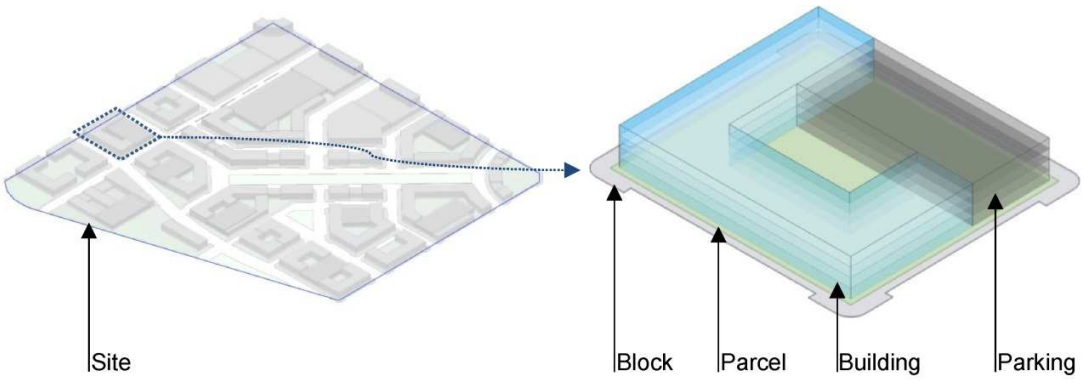


Figure 15. The type of PUDO in the PUDM

information of building topology and regulatory information. In addition, the Building object has parameters controlling its' geometry such as building heights, the number of floors, and building setbacks. The parameters of the Building object represent the urban regulation provisions controlling building form.

The Parking object is either a structured parking garage or an on-site surface parking. The Parking object holds information of parking structure topology and parking regulatory constraints. In general, the Parking standards have provisions regulating the total parking numbers, parking setbacks, and entrance locations. The Parking object corresponds to the Parking standards.

In the urban design code of the City of Ventura (2007), a residential building shall provide one parking space for every 1,500 square feet floor. It implies that the total number of parking can be calculated from the gross floor area of buildings. The parameter values of the Parking object can be decided through complex calculations using multiple values from other PUDOs.

4.3.1.1 Object hierarchy of the Code objects

Five types of Code objects compose the PUDM with the following object hierarchy (Figure 16). As described previously, an object hierarchy is established by hosting objects in parametric modeling. Hosting and hosted relationships are modeled between Block and Parcel, Parcel and Building, and Building and Parking. As shown in the diagram, we can see a Site object hosts multiple Block objects. Such one-to-many relationships can be established among the objects, but we demonstrate one-to-one relationships in this research to simplify the modeling and application development.

In Revit, the host and hosted information is stored in the parameters, so the relationship can be

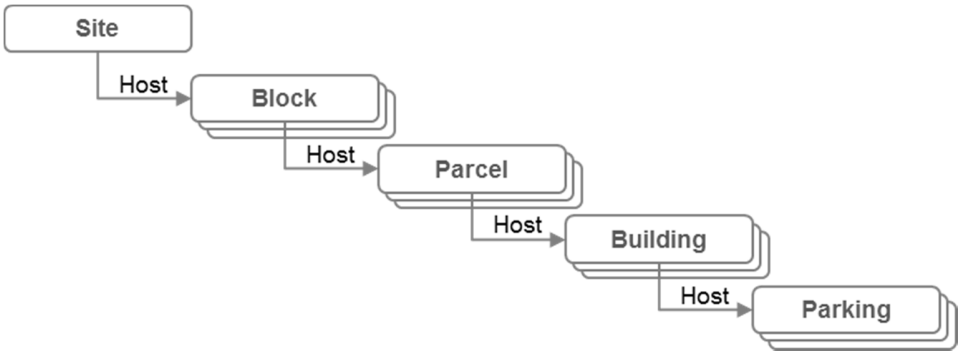


Figure 16. An object hierarchy of PUDOs

retrieved from the parameters. For instance, calculating the open space ratio requires information of the parcel area as well as the footprint area of the building and the parking structure. Generally, PUDM consists of multiple BIM objects. By using the hosting information, relevant objects can be easily filtered from the multiple objects and then required information can be obtained from the filtered objects.

In addition, a consistent naming convention is applied. For instance, a Building object in the lot 001 is named as “L001Building.” “L001” stands the address in which the building is located. “Building” stands the type of the PUDO. In a same manner, the Parking object in L001 is named as “L001Parking.” The consistent naming convention eases the complexity in the application development without the time consuming search process.

4.3.1.2 The parameter types in regulation objects

Each object has parameters for geometry update, calculation, and analysis. Table 6 shows major parameters within the five PUDOs. Generally, parameters of each Code object represent a part of provisions of the corresponding zoning code components. Some parameters are added to establish interrelationships among the provisions. Some parameters are used to store calculated values or analysis results. The type of parameters can be categorized as four groups: spatial

Table 6. Main parameters of PUDM

Object	Parameters	Descriptions	Units
Site	Parameters for the economic analysis are included. See details in Table 16		
Block	Streetscape type	The type of the streetscape	N/A
	Sidewalk width	Width of the sidewalk space	ft.
	Street parking width	Width of the street parking space.	ft.
Parcel	Open space type	The type of the open space	N/A
	Transect code	Zoning codes from rural to urban	N/A
	Open space ratio	The percentage of open space to the property area	%
	Open space area	The lot area excluding structures	ft^2
Building	Building envelope type	The type of the building envelope	N/A
	Setback (BTL/Front)	The distance from BTL to the building envelop	ft.
	Setback (Side)	The distance from the side property line to the building envelop	ft.
	Setback (Rear)	The distance from the rear property line to the building envelop	ft.
	Depth (Main building)	Building depth of the main building	ft.
	Depth (Sub building)	Building depth of the sub building	ft.
	Building Height (Ground floor)	The first floor height from slab to slab	ft.
	Building Height (Upper floor)	The upper floor height from slab to slab	ft.
	Number of floors	The floor numbers of the building	N/A
Parking	Parking type	The type of the parking structure	N/A
	Required parking number	The minimum parking number required to provide	N/A
	Number of street parking	The number of designed on-street parking spaces	N/A
	Available footprint area	Property area exclude buildings and open space	ft^2
	Planned footprint area	The first floor area of the parking structure	ft^2
	Parking space area per car	The parking lot area for one car	ft^2
	Number of floors	The floor numbers of the parking structure	N/A
	Building height	The height of the parking structure	ft.
	Distance from buildings	From the parking structure to the buildings	ft.
	Distance from parcel lines	From the parking structure to the parcel line	ft.

regulatory parameter, non-spatial regulatory parameter, performance (function) parameter, and supplementary parameter.

- Spatial regulatory parameters directly control geometry of PUDOs (e.g., Depth and Height of the Building object). The parameter value can be obtained from the urban

regulation provisions.

- Non-spatial regulatory parameters store the zoning code requirements. These parameters are indirectly related with geometry of the PUDOs. For instance, open space ratio of the Parcel object stores the minimum value of open space ration designated in the Open space standards. The ratio does not directly affect geometry of the code objects, but it affects other geometrical attributes such as the footprint area of buildings and parking structures.
- Performance parameters store performance analysis results of either individual Code objects or entire PUDM. For instance, a Site object has a set of performance parameters to store economic analysis results.
- Supplementary parameters store temporary object information or intermediate values for calculations. These parameters are not explicitly identified in the urban regulation provisions. For example, the Parking object has a parameter to store the available

The Site object has a set of parameters for the economic analysis based on simplified Pro Forma estimation, which is not described in Table 6. Details on the economic analysis through applications are described in Chapter 7.

The Block object has parameters relating the streetscape standards. Dimensions of sidewalks and on-street parking are implemented as parameters.

The Parcel object corresponds to the provisions in the regulation plans and the open space standards. In view of parametric modeling, the Parcel object is not transformed by parameters, so no spatial regulatory parameters are included in the Parcel object. Parameters for transect code is built in the Parcel object, which can affect other Code objects' attributes.

The Building object represents the building envelope standards. Building height, disposition, elements, and use are major sections of the building envelope standard. Parameters controlling building height and location are implemented in the Building object.

The Parking object has parameters for storing regulation requirements, counting required parking numbers, controlling dispositions, and updating dimensions of the Parking object based on the numbers.

4.4 The Parametric Modeling Process of PUDOs

PUDOs visualizes form implication of urban regulations. The geometrical representations of PUDOs are created with 3D solid objects and a set of parameters. The solid objects are designed to transform their geometry upon parameter changes. I formulate the parametric modeling rules and process for the urban regulation modeling, which will be explained in the following section. Following guiding questions are addressed.

- Which geometry compositions can ease the complexity in parametric modeling? Is it applicable for urban model creation in BIM?
- Which geometry compositions can enable transforming geometry by multiple parameters?
- How can object geometry be adjusted to cope with the particular site topology without losing parametric model mechanisms?
- How can behavior formulated from urban regulations be represented with BIM objects and their parameters?

The parametric modeling rules and processes for urban regulation modeling are explained in three phases. First, “Decomposition of a code object” explains the analysis process of

geometrical composition of PUDOs by using the Block object example. The geometry of a PUDO is decomposed with a series of solid and void objects that are called as object elements. Second “Composition of a solid object” demonstrates the modeling process to compose a solid object from multiple object elements without losing parametric modeling mechanisms. I demonstrate how a Building object is constructed from a set of provisions of the building envelope standards such as RBL, setbacks, and building height. Third, “Spatial operations to adjust object geometry” illustrates the modeling methods enabling the assembled objects to express accurate geometry by combining void objects in BIM. I demonstrate how a setback provision can be modeled and applied to the Building object.

4.4.1 Decomposition of object geometry

A PUDO is comprised with a series of elements. Individual elements represent the constraints of the zoning code components. For instance, a Block object represents streetscape such as sidewalks and street parking, so parameters controlling streetscape are built in the Block object (Figure17).

A Block object consists of multiple elements such as sidewalks, curve corner, on-street parking, and parcels. The general rules of geometry composition with individual elements can be formulated from the graphical standards or textual provisions of the urban regulations. Typically, parcels are located within a block. On-street parking is planned along the block boundary. A sidewalk is located in the space between the curb edge and the parcel boundaries.

The dimension of individual elements such as sidewalks and on-street parking can be identified from the urban regulation provisions such as sidewalk width, on-street parking width, and curb radius. Such values are stored in the parameters of individual elements. As parameter values are changed, the element geometry can be updated.

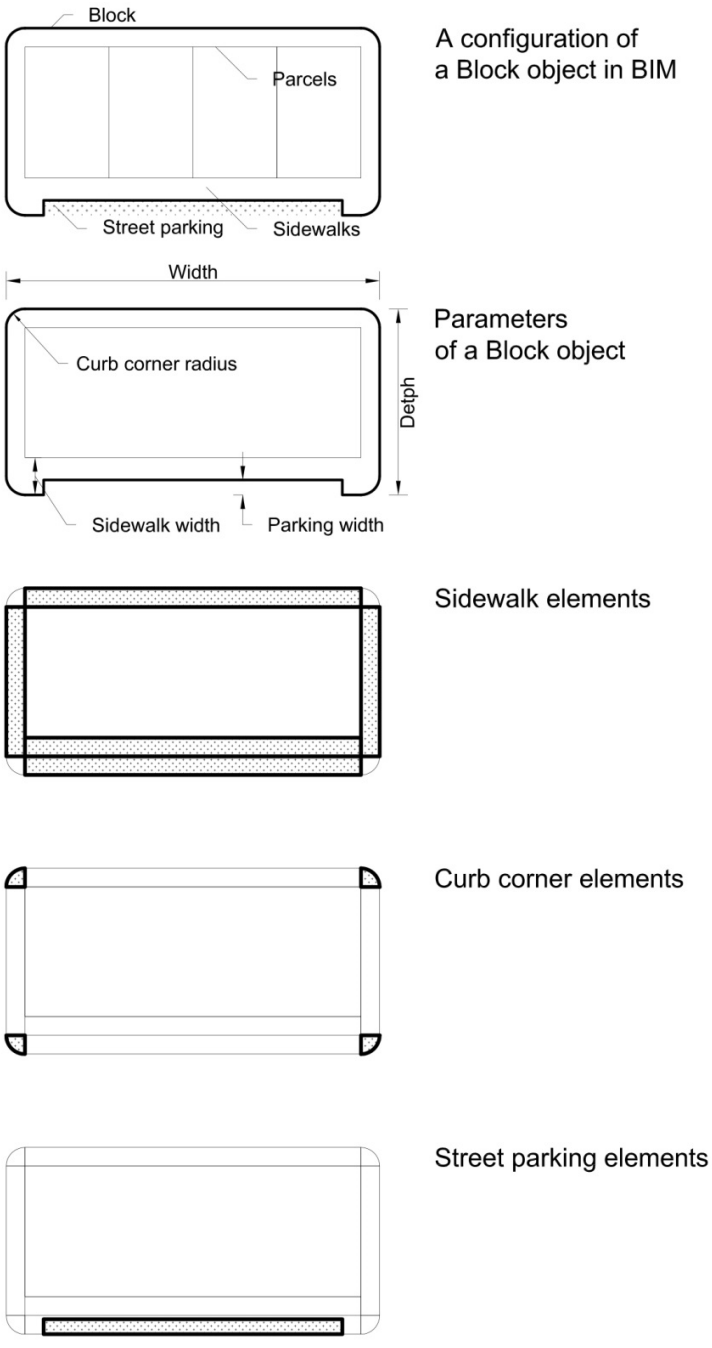


Figure 17. Geometric composition of the Block object

The configuration of individual elements is determined by the geometry compositions of the PUDOs. For instance, if the curb corner element and the sidewalk element are modeled as one, the new element would have different geometry and parameter sets. Therefore, the geometry composition of the PUDOs determines the configuration of individual elements.

In terms of form and behavior of BIM objects, element geometry is form. The way of changing form by parameters is behavior. The way of geometry composition of the Block object determines form and behavior of individual elements.

To transform geometry by various parameters, the Block object is decomposed with a set of solid and void elements. The elements of curb corner, sidewalk, and parcel are created with solid elements. The elements of on-street parking are made with void elements.

The curve corner element is a solid element that represents the block corner space (Figure 18). For the curve radius and the angle, two parameters of CurveAngle and CurveRadius are created. The radius value corresponds to a provision in the streetscape standards. The angle value varies over the block topology. The Sidewalk element corresponds to the sidewalk space that is governed by the streetscape standards in the urban regulations. The width of sidewalks is modeled with the SidewalkWidth parameter. The on-street parking element is a void object that cuts out the parking spaces from the sidewalk element. The dimension of street parking space is controlled by the OnStreetParkingWidth parameter, corresponding to the parking requirements of the streetscape standards of the urban regulations.

The example demonstrates the Block object only, but similar modeling rules are applied to other PUDOs including Parcel, Building, and Parking.

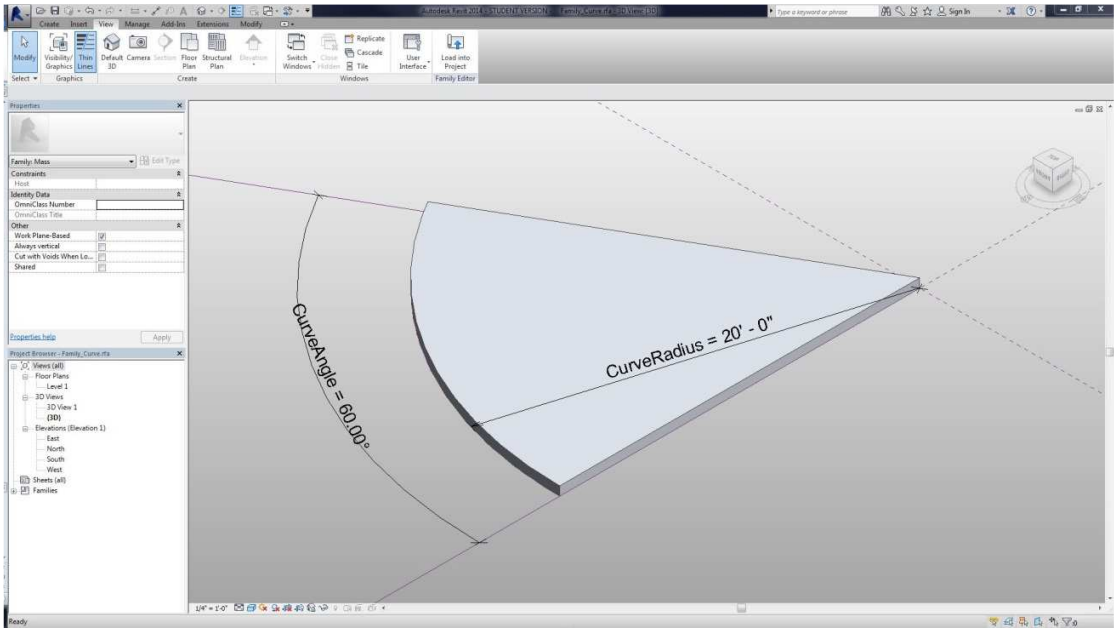


Figure 18. A curve corner element in the Block object

4.4.2 Composition of object geometry

The Building object is a mass in BIM corresponding to the building envelope standards in the urban regulations. One of the significant provisions in the regulating plan is the BTL or RBL that designates the front façade location. This section describes how the RBL requirement is parameterized in the PUDOs.

Figure 19 shows a parcel having a BTL on the curved property line. The example illustrates the modeling process of a Building object from the RBL, setback, building height, and building depth. In the first diagram, the BTL is located in the bottom side of the parcel. Other sides are controlled by the setback regulations.

The shape of the RBL reflects the parcel geometry. In this example, the bottom side parcel boundary has four line segments and they work as the RBL. The step 2 is to create four mass

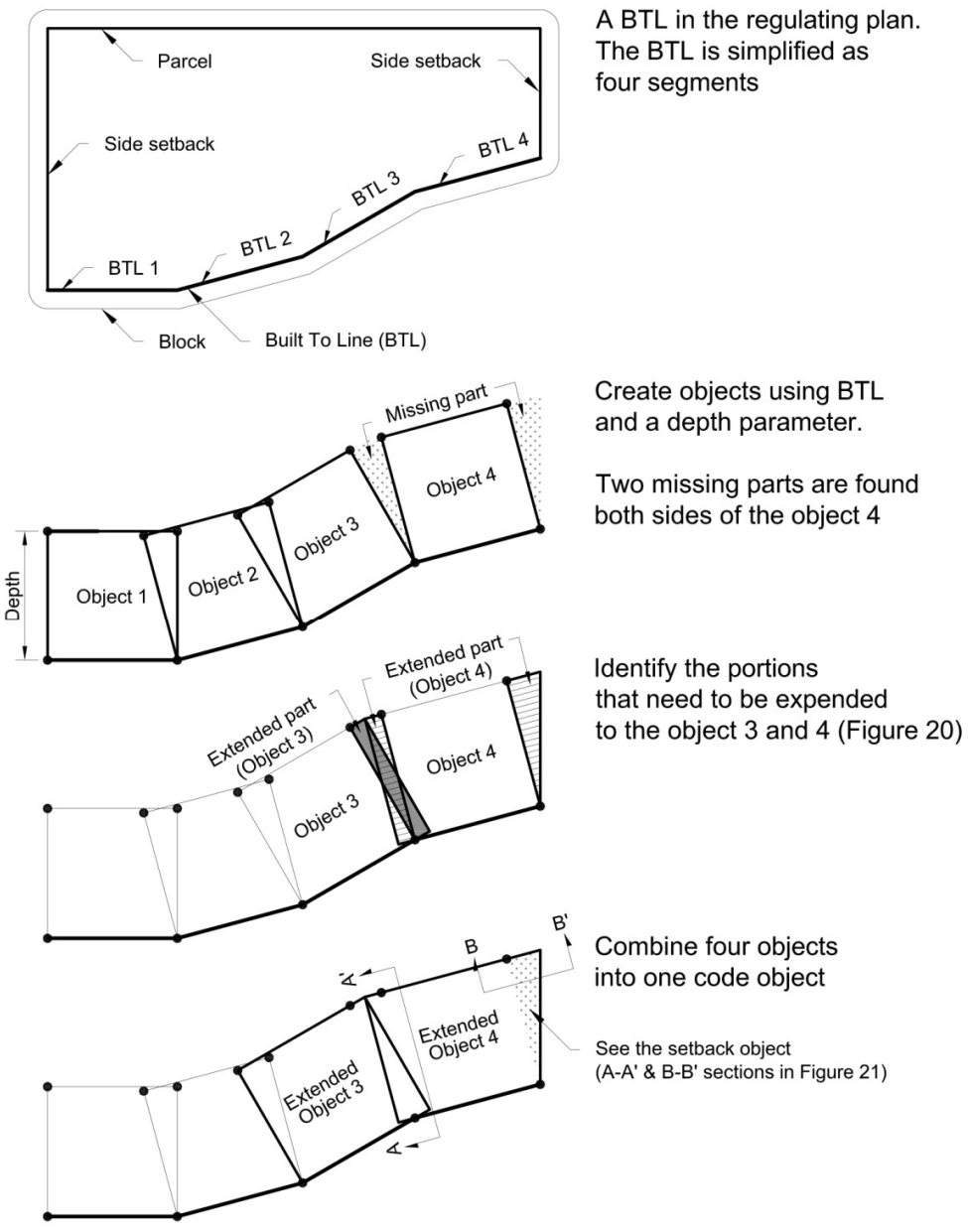


Figure 19. A modeling process of a Building object from the RBL

objects using four segments. The arbitrary values are applied to the depth parameter. As shown in the second diagram, two missing parts are found both side of the object 4, while the object 1,

2, and 3 are overlapped respectively. Three overlapped mass objects can be joined in Revit.

The two missing parts are modified in the step 3. The object 3 and 4 are stretched until they become overlapped. To apply the side setback rule to the object 4, the right-side edge of the object 4 is stretched to cover the missing part.

In sum, the RBL regulation and the building depth are applied to four objects. The side setback rules are applied to the object 1 and 4. The object 3 and 4 are modified to be joined. The object 4 is stretched to apply the side setback rules.

Based on this example, I formulate the general modeling process of the Building object as follows:

(1) Define the RBL from the Parcel object (step 1).
(2) Decompose the line segments of the RBL and create individual objects (step 2).
(3) Identify the gap between two objects and stretch them until the gap is covered (step 3).
(4) Identify the gap between an object and the setback line. Then stretch the object until the gap is covered (step 3).
(5) Join individual objects into one Building object (step 4).

Through this process, a Building object can respond to RBL, setbacks, building height, and building depth.

4.4.3 Spatial operations to adjust object geometry

One remaining question is how much the objects need to be elongated in the step 3 and 4. The dimension of a mass object can be manipulated in two ways: One approach is manipulating dimensions of the solid objects based on equations and the other approach is manipulating geometry of the solid objects with void objects.

4.4.3.1 Manipulating geometry with equations

The equation-based parametric modeling can produce accurate object geometry. In parametric BIM, an equation can make an association among parameters. Once an equation is written in a parameter, an accurate parameter value can be calculated by equations. Figure 20 is a close-up of Figure 19 to show the missing part between the object 3 and 4. W3' and W4' are dependent on the angle between two objects, relating to the parcel geometry. A simple calculation using the following variables and equations can return accurate values to be stretched as follows:

- W3: the original width of the object3
- W4: the original width of the object4
- D3: the depth of the object3
- D4: the depth of the object4
- θ: The angle between two objects
- $\mathrm{W3}' = \mathrm{W3} + (\mathrm{D4} - \mathrm{D3} \times \cos\theta)/\sin\theta$
- $\mathrm{W4}' = \mathrm{W4} + (\mathrm{D3} - \mathrm{D4} \times \cos\theta)/\sin\theta$

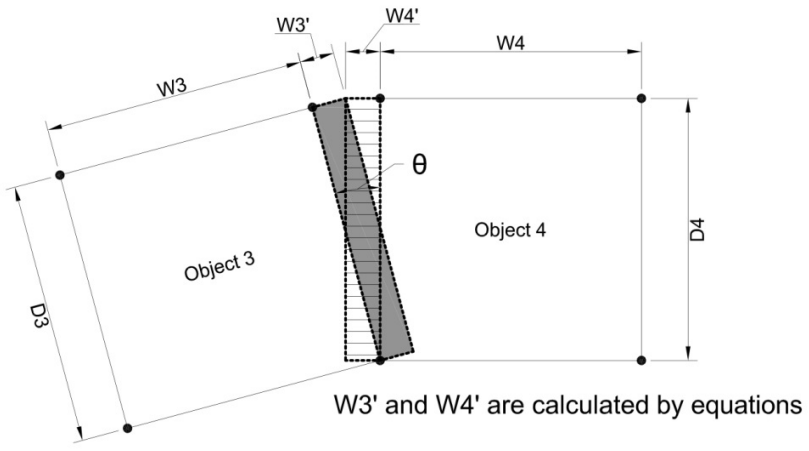


Figure 20. Required variables for the equation-based geometry manipulation

The calculation is straightforward, but such variables need to be defined before the calculation process, and the calculated values need to be applied to the corresponding solid objects. To apply the equations into parameters, such variables need to be defined as parameters during the modeling process. For instance, to stretch two objects, five parameters need to be assigned to two objects. If a Building object has more missing parts, more parameters would be assigned during the modeling process.

This approach can yield accurate results, but it can delay the parametric modeling process for the urban model having multiple objects.

4.4.3.2 Manipulating geometry by hosting void objects

A void object can affect the geometry of a solid object. In parametric modeling, the disposition and the dimension of a void object can be manipulated by object parameters so that a solid object can represent diverse geometries by combining with a series of void objects.

Manipulating geometry of solid objects with void objects in parametric modeling can ease complexity in complex geometry creation. Often, changing object's location via parameters in parametric modeling is more straightforward than changing object's geometry through parameters. When creating complex form, for instance, multiple parameters are often built in an object and an attribute of object geometry can be manipulated by multiple parameters, which can make the parametric modeling process complicated.

The following example in Figure 21 demonstrates how combining solid and void objects can create the same geometry with the object driven by equations in Figure 20. The same Building objects in Figure 18 and 19 are used again for clear understanding.

The first diagram shows the missing portion between two objects, which will be filled with solid

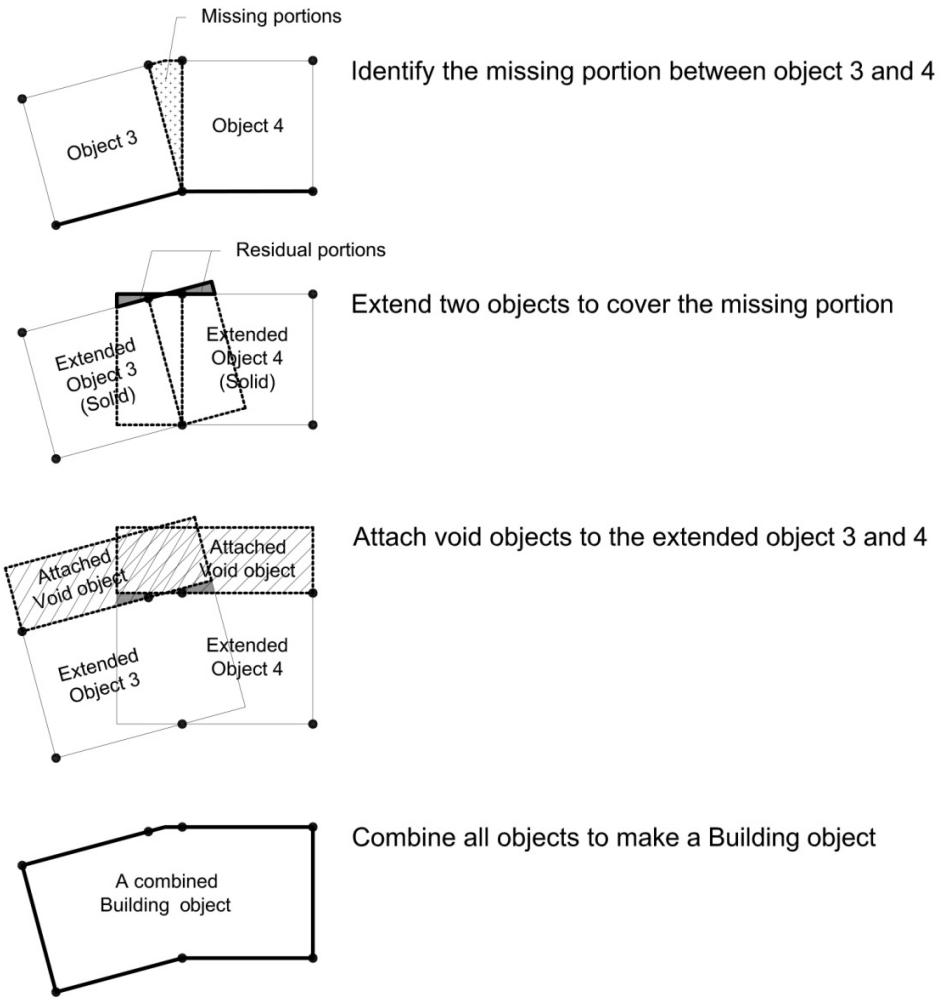


Figure 21. Adjust solid object geometry with void objects

objects. Identifying the portion is the first step. Second, two objects are stretched until the missing portions are covered. There is no need to measure the accurate dimension. As two objects are extended and become overlapped each other, two residual portions are generated in the second image. Third step is to attach two void objects to the upper boundary of the extended object 3 and 4 where two residual portions are remained. Width and depth of the void objects can be roughly assigned until the residual portions are covered. There is no need to use accurate

dimensions. Finally, individual solid and void objects are combined to create a Building object having a proper geometry.

Without calculations for accurate parameter values, objects geometry can be manipulated in the second and the third steps. The position of the objects is precisely assigned, but the dimension of objects is not critical in this approach; only if void objects are placed correctly to completely cover the residual portions of the solid objects, the residuals can be hidden in the Building object. No matter how large the void objects are. Under this beneficent capability of parametric modeling, combining solid and void objects can generate the identical object geometry with the equation based parametric modeling process.

In sum, a series of solid and the void objects compose PUDOs. The solid object is created with a simple geometry and their parameters control basic dimensions such as width, depth, and height. The void object is created with parameters controlling its location and dimension. The void objects are hosed by the solid objects to modify the solid object geometry. The parameters of the void objects change the location of the void objects.

4.4.3.3 Modeling provisions by nesting void objects

This section explains the modeling process of setback provisions by nesting void objects. The same Building object in Figure 18~20 is exemplified again. Setback requirements, for instance, affect the building geometry, so they are built in the Building object.

The setback regulation controls the distance from the property lines to the structure, which can determine the façade location of buildings and parking structure. In the urban regulations, various types of setback regulations are designated. In the Building object, front, side, and rear setbacks are implemented. Setback regulations are implemented by two approaches as shown in

Figure 22; changing the solid object location for front setback (A-A' section) and changing the void object location for side and rear setbacks (B-B' section).

Two sections show the simple solid and void objects in Revit. Grey colored objects are solid objects and the hatched objects are void objects. Dots represent reference lines in Revit that can host other reference lines or objects. The distance between reference lines can be controlled by parameters. An object can be hosted either by a reference line or another object's surface. As the host object changes the position, a hosted object moves accordingly.

In parametric modeling, behavior for setback regulations is implemented in two ways. First, the

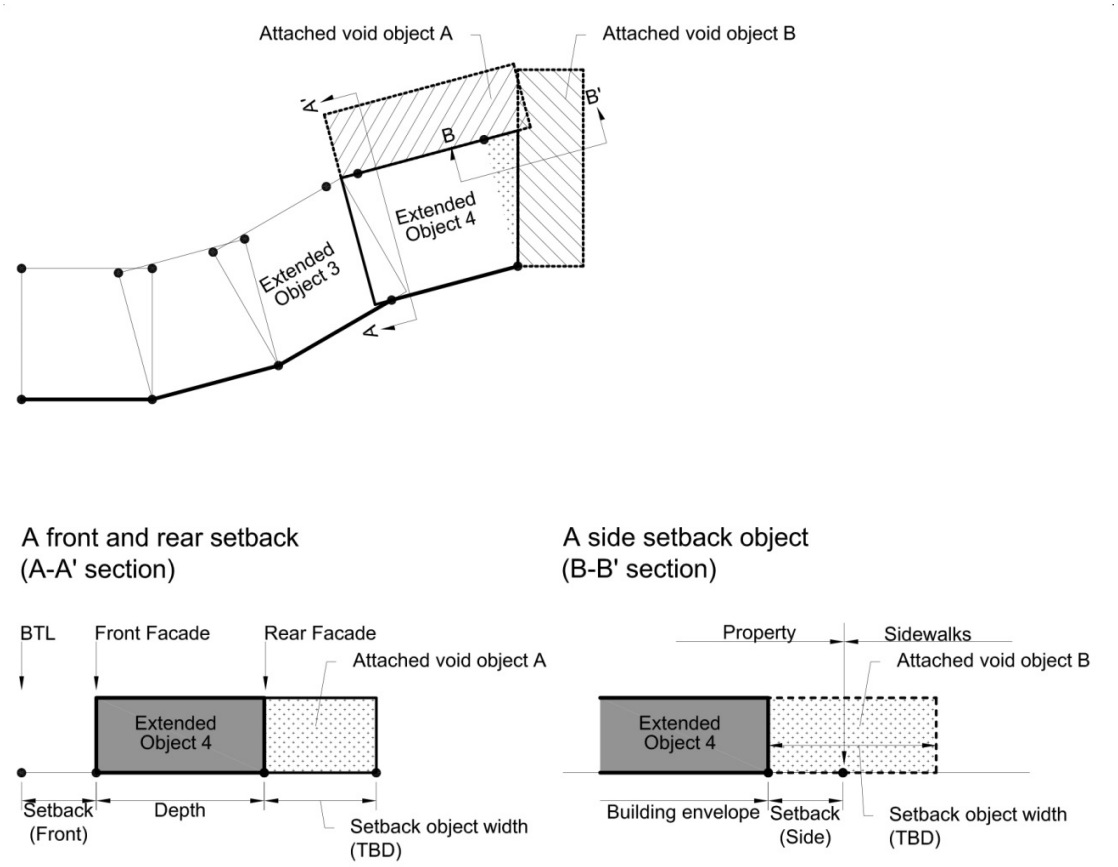


Figure 22. A composition of the setback object

front setback moves the front façade location. In the A-A' section, the second dot from the left is the reference line A, the reference line A hosts the extended object 4, and the object 4 hosts a void object A. As the reference line A moves by parameter, the object 4 moves accordingly. The attached void object A cleans up the residual portions near the rear façade described in figure 20. Second, the side setback cuts the building mass by moving the void objects B. In the B-B' section, the side setback moves the reference line B, the reference line holds the void objects B, and the void object cuts the object 4. The geometry of the side setback reflects the site topology. The location of the void object B needs to be accurate but the void object dimension is not critical.

Since the setback values can be manipulated by parameters, individual Building objects enable to represent the wide range of setback values. In addition, the geometrical representation of setback rules responds to both regulation provisions and the parcel topology because a Building object is created on the regulating plan.

4.5 Summary and Discussions

In this chapter, I described the structure of PUDM and PUDOs. The structure of PUDM follows the structure of zoning code components defined in Chapter 3. The object hierarch of PUDM represents the relationship among the zoning code components. The type of PUDOs corresponds to the type of zoning code components.

Then I explained the parametric modeling process for urban regulation modeling in parametric BIM. For proper geometry representation of urban regulations, a PUDO is decomposed with elements, the individual elements are combined, and the geometry of the combined elements are adjusted by using the parametric modeling capabilities in BIM.

Even though I explained the modeling process with examples in detail, I could derive significant rules in urban regulation modeling in Parametric BIM.

4.5.1.1 Form driven by the code provisions

The geometric representation of urban regulations is implemented with the parametric modeling method in BIM. The object geometry does not always represent building design that will be built. Buildings may be much more complex or diverse than the form driven by the code provisions. However, some prescriptive zoning provisions directly reflect building design. For instance, the RBL determines the front façade location of the Building object. The requirements of building height or the number of floors are often directly applied to building design.

4.5.1.2 Various parametric modeling processes for a zoning code provision

As I described, a BIM object can be understood with form and behavior. In PUDOs, form is a geometrical representation of urban regulations and behavior is a way of geometry transformation by parameters. The type of behavior determines the relationships between form and parameter. Therefore, the type of object behavior is related to the parametric modeling process.

A regulation constraint can be parameterized with a set of object behavior. As shown in Figure 20, a setback provision was implemented in two ways. First, the building façade was moved to the designated position. Moving the façade can be behavior of the BIM object. Second, a void object was moved to cut a building mass. Moving a void object can be a behavior.

In short, object behavior affects the parametric modeling process. In addition, a set of object behavior can be formulated from a regulation constraint. Therefore, a regulation can be parameterized in various parametric modeling processes.

4.5.1.3 Varying complexities in the parametric modeling process

A parametric modeling process can affect the level of complexity. I explained a setback provision can be parameterized in two ways: moving an object face and moving a void object.

To move a face with a parameter, the parameter needs to be linked to the face. When multiple faces are controlled by various setback regulations, corresponding parameters need to be related the relevant faces. Consequently, the number of parameters of a BIM object can easily increase.

To move a void object, a parameter needs to be linked to the reference point or line in Revit. The parameter does not change the void geometry. For multiple setbacks, adding void objects is required. By moving void objects, the shape of the mass object can vary in the user interface without modifying the original solid object.

In sum, transforming geometry using parameters is more complicate than changing an object location using parameters. Two approaches represent the same provisions, but they pose subsequently different complexities in the parametric modeling process. In Parameterization of urban regulations in Parametric BIM, behavior of BIM objects is a critical determinant of complexity of the parametric modeling process.

4.5.1.4 Interrelationships among code provisions are established by the applications

Interrelationships among code provisions are often discovered from the conditional or the comparative regulations. For instance, maximum height of the parking structure is often limited by height of main building. The number of parking spaces can be calculated using the building floor area. In terms of PUDOs, two cases pose interrelationships between a Building object and a Parking object.

In Parametric BIM, such relationships may be established by equation-based parameters. If an

equation is written in the height parameter of the Parking object, it may be possible to relate two objects. However, a BIM object is not too intelligent to automatically access other BIM objects. Even if the building floor area increases, the Parking object in BIM cannot recognize the changes in other objects. Furthermore, to write an equation in the parameter, required information is to be stored during the modeling process. Whenever such equations are to be edited, individual PUDOs need to be modified. Therefore, adding equations in the parameters can delay the modeling process.

For such an interrelationship, I create PUDAs using Revit API, which perform continuing parameter change and dynamic model update. The applications enable reading required information from any Code objects, calculating new values from object information, and feeding calculated values into parameters of PUDOs. The application development process will be explained in the next chapter.

5 DEVELOPMENT OF PUDA

In this chapter, I describe the prototyping process of the PUDAs that establish relationships among the PUDOs and their parameters. The applications have been developed under three scopes: (1) representing compound matrixes among zoning code components and their provisions as the relationships among PUDOs and their parameters; (2) performing complex operations commonly discovered in use of urban regulations such as the overlay process of the multiple zoning code components; and (3) enhancing the modeling process of the PUDM. The prototypes are developed with the BIM API, Integrated Development Environment (IDE), and an OOP language, C#.

In the following sections, I will investigate where the matrixes exist and how such matrixes can defer the use of urban regulations. Then I will describe which types of the application prototypes will be developed in this research. Lastly, the individual applications will be described in three folds. In view of the application operators, how to use the applications is explained. In view of the software development, how to design and create the applications is described. For future development, potentials, challenges, and limitations are discussed.

5.1 Matrixes among the Code Objects

As I identified earlier, a zoning code consists of a set of code components, the code component consists of sections, and the section is comprised with a series of provisions. In Chapter 5, five types of code components are categorized including the regulating plan and four standards of streetscape, open space, building envelope, and parking. Then they are represented as the five types of Code objects in BIM including the Site, the Block, the Parcel, the Building, and the Parking objects. The relationships among the five zoning code components are represented with

an object hierarchy among the Code objects.

The matrixes are often discovered in the provisions of the urban regulations. When a provision in one zoning component is related to other provisions within other zoning code components or when a provision is related to multiple provisions, such matrixes among provisions cannot be easily embedded in the PUDM.

The provisions of the urban regulations are represented as attributes of the PUDOs, which can be accessed through parameters of individual objects in BIM. When a parameter in one object is dependent on other parameters, the relationships are established by the PUDAs.

5.1.1 Where the matrixes exist

The range of provisions showing matrixes is too broad to be completely implemented in this research context. However there are general rules about the matrixes across the zoning code components and the PUDOs (Table 7), which can inform the prototyping scope for the PUDAs.

The first column shows the zoning code components and the first row shows Code objects in BIM. When a regulation in one zoning code component is related to Code objects, such relationship is marked in the table. For example, the on-site open space regulation in the Open

Table 7. General relationships among zoning code provisions and PUDOs

Zoning codes		PUDOs				
Components	Provisions	Site	Block	Parcel	Building	Parking
Regulating plan	• Transect	O	O	O	O	O
	• RBL		O	O	O	O
Streetscape standard	• Road configuration	O	O	O	O	
Open space standard	• On-site open space			O	O	O
Building envelope standard	• Disposition		O	O	O	O
	• Height				O	O
Parking standard	• On-site parking		O	O	O	O

space standards is associated with the Parcel, the Building, and the Parking objects as follows: (1) the Parcel object affects the minimum open space area and disposition of open space; (2) the Building object constraints the available location and the area for open space; and (3) the Parking object influences the usable land location and the area, while the area of the parking object is associated with the Building object.

The object hierarchy is related to the level of complexity in the interrelationship. In the first column of the table, the higher level in the zoning code hierarchy is listed from top to bottom. In the first row of the table, the higher level in the Code object hierarchy is placed from the left to right. Often provisions in the lower level are dependent on others in the upper level. In that reason, matrixes exist in the provisions of the Building envelope standard and the parking standard. For instance, the parking structure capacity is determined by the building scale, the height of the parking structure is limited by the main building height, and the footprint area of the parking structure is restricted by the building footprint area and the open space area. In these provisions, the matrixes are found across the Code components.

On the other hand, some matrixes are discovered within a code component. When a provision is dependent on others in the same code component, the interrelationship can be understood without considering other code components. In view of the Code objects in BIM, this type of matrixes can be established within a Code object, which is less complicated.

In general, similar relationships are identified among the PUDOs in BIM in that the Code objects represent the zoning code components. As the first column of Table 8 is replaced with the Code objects, the relationships among Code objects can be understood, which is more relevant to the application development. The five types of the Code objects are listed in the column and the row respectively in the following table.

Table 8. General matrixes among the PUDOs

	Site	Block	Parcel	Building	Parking
Site	O	O	O	O	O
Block	O	O	O		
Parcel	O	O	O	O	O
Building	O		O	O	O
Parking	O		O	O	O

As shown in the dot line box, the Parcel, the Building, and the Parking objects are associated each other, so the use of them can be complicated. The Site, the Block, and the Parcel objects are also associated each other, but they are less dynamically changed that the Building and the Parking objects. For instance, topologies of the Site and the Block objects are defined in the regulating plan. Subdivision information for the Parcel objects is often defined prior to the design process of the building and the parking structure.

5.1.2 How the matrixes can delay the use of urban regulations

To identify the application development scope, it is significant to understand when and how the matrixes among the code provisions can impede the use of the urban regulations. In general, the level of complexities varies over the urban regulations, but it is also related to the scale of the urban model. The matrixes among the provisions make the regulation interpretation complicate. If one provision is associated to multiple provisions, it is not straightforward to determine an interpretation sequence. Even if a certain interpretation order is established, applying the interpretation process to multiple lots may become time consuming.

The following examples describe the common problems caused by the matrixes of the urban regulations and the large scale of the urban models. Addressing each problem will be an

objective of the Parametric Code application development.

5.1.2.1 Difficult to locate required information

When a provision is constrained by others, it is difficult to locate the required provisions from a series of zoning code standards. One zoning code I reviewed has five chapter and about 300 pages. Applications can be developed to automatically access multiple PUDOs, filter the relevant objects, and locate the required parameters.

5.1.2.2 Difficult to comply multiple requirements that are associated each other

When a provision is constrained by others that are not fixed (e.g., a minimum, a maximum, or a range value), it is required to wait until others are decided. When any changes on others, the provision dependent on them needs to be updated accordingly. It can make the use of the provision complicate. For instance, the open space ratio can be calculated after the parcel, after the building footprint and the parking structure footprint are decided. Applications that can locate the associated parameters from the multiple objects and manipulate them simultaneously will resolve this issue.

5.1.2.3 Difficult to review all

A typical zoning code covers a series of districts, so checking regulation compliance for multiple developments required time and effort driven processes. Applications can be developed to automatically perform a series of code compliance checking throughout the whole model when the code checking rules can be defined in the applications.

5.1.2.4 Difficult to create the urban model

Due to the scale of the urban regulations, an urban model has multiple parcels, blocks, or districts, which can delay the modeling process. Applications that can create an urban model or

assemble the individual PUDOs will relieve this difficulty.

5.1.2.5 Difficult to test a set of provisions

In a same reason, testing a set of provisions takes substantial efforts. Applications can be designed to access individual PUDOs and update the object attributes via their parameters. Performing a series of parameter updates enables a set of provisions to be tested by the application.

5.1.2.6 Difficult to identify the effect of urban regulations on the built environment in the regulation making process

Due to the above reasons, creating urban models from urban regulations is difficult, which prevents evaluating or simulating the effect of urban regulations on the built environment via computer tools (check, avoid overstatements, or limit to the physical attributes of the urban model). In the code making process, it can make difficult to answer the questions such as how a building height incentive affects the energy consumption or how a setback distance affects the construction costs. Applications that can convey performance analyses using the PUDM such as an economic and an energy performance analysis will be developed. The applications may provide various performance analysis results in the code making process.

5.2 Prototypes of the PUDAs

To that extent, a set of prototypes has been implemented under the application development objectives (Table 9). Each prototype demonstrates how such difficulties are addressed by the PUDM and the PUDAs. The following section describes the methods and tools in prototype development.

Table 9. Prototype of the PUDAs

Prototypes	Definitions
Task: Generating a Parametric Urban model	
• GeneratePlan	Assembles individual Code objects to create a Parametric Urban model
• GenerateArray	Creates object arrays of plants and lighting fixtures on the Block objects
• UpdateArray	Edits the object types and spacing in arrays
Task: Assigning transect codes	
• AssignTransectCodes	Updates transect code types and visualizes transect codes in the Parametric Urban model
Task: Testing parameters	
• AssignBuildingCodes	Modifies the parameters of the Building objects and update the Parametric Urban model
Task: Editing regulatory variables	
• AssignParkingCodes	Modifies the regulatory requirements in the Parking objects
Task: Updating models through (code) matrixes	
• UpdateParking	Calculates the required parking numbers, estimates the available parking structure geometry, and updates the entire parking structures
Task: Checking code compliance	
• CheckCollision	Checks collision among the Code objects, reports the checking results, and colorizes the problematic objects in the Parametric Urban model.
Task: Visualizing performances	
• VisualizeDensity	Measures density of individual parcels, calculates the average density, and colorizes deviations of individuals from the average in the Parametric Urban model
Task: Reporting model information	
• ReportModelInfo	Updates a set of parameters, captures model information and the floor area ratio of each set, and generates reports
Task: Analyzing performances	
• ReportProForma	Updates a set of parameters, performs a series of Pro Forma estimation, and generates Pro Forma reports

5.3 Methods, Tools, and Techniques

5.3.1 Parametric modeling

The parametric modelling capabilities in BIM enable users to assign new parametric values from the user interface. However, some associations across multiple Code objects cannot be established by parameters in that one object cannot intelligently recognize information within others. For example, Floor Area Ratio (F.A.R.) is the ratio of the total floor area of the buildings to the property area.

To obtain this value from the parametric PUDM, two values are needed; gross floor area can be read from the building mass and the property area can be read from the parcel model. Another example is the maximum building footprint area. In general, a certain ratio of the property area is allowed for the building footprint area.

To do so, the property area and the allowed ratio need to be collected from the parcel model, and it can be transferred to the building mass. However, single mass model in Revit cannot read parameters from other mass models.

In this circumstance, such associations across the entire code model can be established by external applications written by using Revit API and OOP in C#. This chapter describes how our developing applications can access the information of the PUDOs, formulate new values using multiple parameters, and transfer the new values to the parametric code projects.

5.3.2 BIM API

- Accessing object information in BIM
- Performing custom calculations using object information
- Automating rule-based operations

- Creating a user-friendly software interface
- Importing and exporting object information of the PUDM

5.3.3 Object-Oriented Programming

Procedural programming or simple scripting often consist of sequence of operations on data during the program executions. However, such linear and sequential data operations can be difficult to manage with increasing complexity.

To address the complexity, OOP was developed to allow source code to be componentization and reused. In OOP, a class is a type that can describe an object representation in that individual objects are created using the classes. A class determines object's common characteristics, attributes (properties), and behaviors (methods). An object in OOP is an instance of a class. It encapsulates both data and behavior. Following definitions would further enable clear understanding on the application development process.

- Namespace is a method to organize classes in different libraries to avoid conflicts among multiple versions of the classes.
- Attributes declare behavior of the command.
- A method performs a particular task within a class.
- A variable is a container that can assign a value to a parameter.

A formal development process in this research is (1) understanding activities in the use of urban regulations, (2) identifying compound relationships among regulation components, and (3) representing the matrixes among regulation components and provisions

5.4 GeneratePlan for Generating the PUDM

GeneratePlan is an application to assemble a series of Code objects to create the PUDM. During

the assembling process, individual Code objects are nested according to the Code object hierarchy and then a series of the Code objects can be grouped automatically. The GeneratePlan application can save modeling time in the urban scale project.

5.4.1 Description

The GeneratePlan application runs without user interfaces. Once the icon is clicked in the Revit menu bar, the application performs the assembly process. To run the application, the predefined Code objects need to be stored in the following folder location.

- Code objects are stored in C:\ParametricCodes\ParametricUrbanModel\CodeObjects.
- The Block, the Building, the Parcel, and the Parking objects are stored under the "CodeObjects" folder respectively.
- Four sub folders are included under the CodeObject folder respectively; Site, Block, Parcel, Building, and Parking. E.g., the Block objects are located under the folder, C:\ParametricCodes\ParametricUrbanModel\CodeObjects\Block.

When other folder structure is used, corresponding file locations should be stored within the application.

5.4.2 Application structure

Once a user clicks the application icon in the Revit menu, the application loads five types of the Code objects sequentially; Site, Block, Parcel, Building, and Parking. The loading process is performed parcel by parcel by following steps.

(1) *Initializing the document*: defines the Revit model, filters, and reference plans. Custom filters are created to find the reference level and required Code objects.
(2) *Loading the Site object(s)*: inserts the Site object into the PUDM.

(3) *Loading the Block object*: inserts the Block objects.

(4) *Placing the Parcel object*: nests the Parcel object on the top face of the Block object.

(5) *Placing the Building object*: hosts the Building object on the top face of the Parcel object.

(6) *Place the Parking object*: hosts the Parking object on the vertical face of the Building object.

(7) *Adjusting parameters*: gives the Code objects have default parameter values.

(8) Repeat (3) ~ (7) for the last of the Block objects.

Nesting one Code object by others is implemented by the application to establish an object hierarchy. Once a Code object is nested or hosted by others, the relationships can be identified as the "Work Plane" parameter. For instance, L001Park is nested by L001Bldg, the value of the "Work Plane" parameter of the L001Park is "L001Bldg: L001Bldg." The parameter value can be read by applications to track the associations across the Code objects.

To nest one Code object by another, following default methods and custom methods are used (Table 10). The LoadFamily method reads the name of the Code objects and then filters the relevant Code objects from the folder. The NewFamilyInstance method has 12 overload methods

Table 10. Major classes and methods of the GeneratePlan application

Classes and methods of the RevitAPI	Type	Description
Document.NewFamilyInstance	Method	Inserts a new instance of a family into the document.
Custom classes and methods	**Type**	**Description**
GeneratePlan.LoadFamily	Method	Inserts a Code object into the PUDM.
GeneratePlan.PlaceFamilyByFace	Method	Hosts a Code object on the face of the other objects using the NewFamilyInstance method.
GeneratePlan.AdjustParam	Method	Initializes parameter values of the Code objects

and one of them is used in the PlaceFamilyByFace. To do so, the hosted family name and the hosting face information are required, so custom functions to identify the information are implemented in the PlaceFamilyByFace method.

5.4.3 Potential and limitations

To place a Code object on the face of other objects, it is required to identify precise coordinate information in the application. Currently, a set of functions are included in the PlaceFamily and the PlaceFamilyByFace methods using the normal vector of the object face. However, the function can work under certain assumptions. For instance, when the Building object is placed on the Parcel object surface, the top surface of the Parcel object is assumed as flat. When the Parking object is placed on the Building object surface, the vertical surface of the Building object has not to be tilted. For the sloped site and tilted building envelope, further developments are required.

5.5 GenerateArray for Designing the PUDM

Array generator is an application to create a series of components in the public space. The urban design components such as trees, lighting fixtures, public furniture can be placed in the open space via this application. In urban design, these components often need to be located along streets such as sidewalk space, transit roads, and building envelope boundaries. When street trees are placed every twenty foot along the major traffic roads, the position and the total number of trees are required information; the position is related to the traffic road location and the tree number is related to the length of the Block object adjacent to the traffic roads. The application identifies the tree array location from the Block object, measures the length of the tree array, and calculates the number of trees within the array. In the system, plant, street lightings, and public space furniture have been implemented via similar or same approach and processes, so user

interfaces and applications of PlantGenerator are explained in this chapter.

5.5.1 Description

The application is developed based on the Revit array function. Figure 23 shows the user interface of the PlantGenerator that generates tree arrays along curve side. As the Revit array command requires user inputs of the array position and the number of array members, the application collects following information from the user interface.

- The distance from the traffic lane to the array: The edges of the Block object are used as reference lines of the traffic lanes.
- The type of the array members: The tree type is asked in this example.

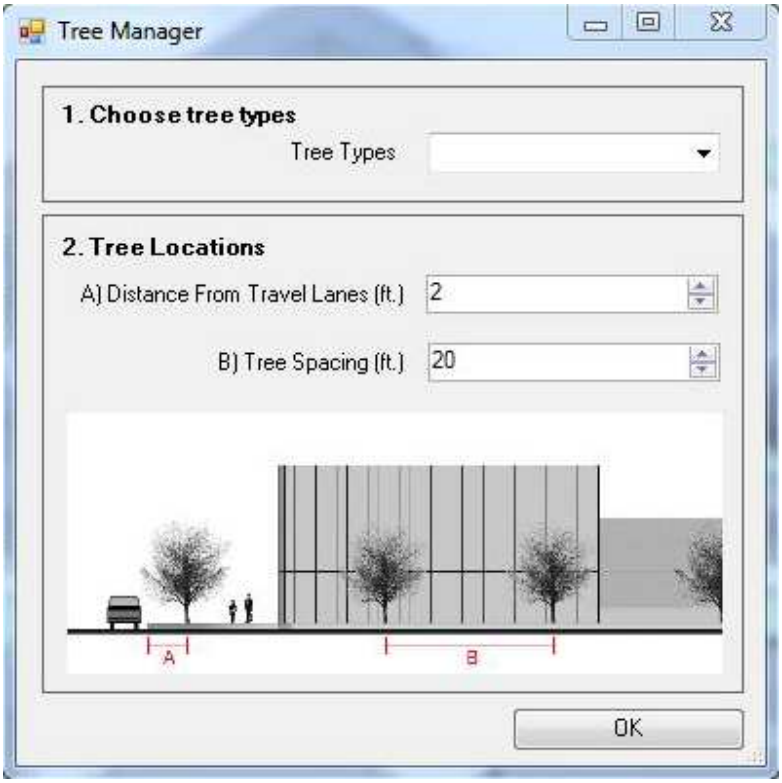


Figure 23. A user interface of PlantGenerator

- A minimum spacing (distance) between array members: This value is used to calculate the required number of array members of each array.

The interface inquires the system operators to select tree types first. Currently, over twenty tree types are implemented, and additional types will be added. Next, the interface asks the distance from travel lanes to the tree array that affects the location of the tree array position. Then the distance between trees are asked to decide the required tree numbers in one tree array. Three user inputs are passed to the main application and then the application automatically generate tree arrays at the entire urban models.

5.5.2 Application structure

From the user interface, a tree type, a distance from the travel lanes to the tree array, and the tree spacing values are obtained. This chapter describes how the application generates an array of trees in the public space based on the three values. A brief workflow, native and custom classes and functions, as well as parameters and variables are explained.

The PUDM consists of five Revit objects: site, block, parcel, building, and parking. The Site family is a wireframe object of the development boundary that includes traffic lanes, open spaces, and urban blocks. The Block family represents the area that is surrounded by streets; it represents public opens spaces, pedestrians, and individual properties. The Parcel family is a subdivided lot within an urban block. The building family represents building masses within the individual lots. The parking family represents either surface or structure parking within the individual properties.

The application generates trees and lighting fixtures in the pedestrian zone of the Block family, so I create a custom filter to collect the Block families from the entire model. The application

performs following six steps.

- *Initializing document*: defines the Revit model, filters, and reference plans. Custom filters are created to collect (1) required Revit families from the urban model, (2) reference lines from the collected families to decide array positions, and (3) existing but unrelated arrays in the urban model for error handling.
- *Reading user inputs from the form*: passes user inputs from the user interface to the application. The application reads three values of the tree type, tree spacing, and the distance from the Block edge to the tree array.
- *Reading reference faces of the Block families*: defines the array position on the surface of filtered Block families. A Block family has multiple surfaces, so I create a custom functions to read all surfaces of the solid model, collect the upper surface that will hosts tree arrays, and then create reference lines on the surface for the tree array generation.
- *Loading family*: places a family instance from the Revit library to the document according to the user selection.
- Creating arrays: generates arrays at the reference lines. Two user inputs of tree spacing and the distance from the road are used to calculate the final array position and the number of array members. In this process, a series of arrays are automatically generated on all properties within the PUDM.
- *Cleaning up*: deletes temporary lines created by the application.

To perform the above tasks, I use native classes and functions Revit provides via RevitAPI.dll and RevitAPIUI.dll. Custom classes and functions are created as follows (Table 11).

Table 11. Major classes and methods of the Array Generator application

Classes and methods of the RevitAPI	Type	Description
Autodesk.Revit.DB.GeometryObject	Class	A base class for all geometries
Autodesk.Revit.DB.Solid	Class	A 3d solid of a geometry
Autodesk.Revit.DB.Face	Class	A face of a 3d solid
Autodesk.Revit.DB.Edge	Class	An edge of a 3d solid
Face.Project	Method	Projects the specified point on the face
Autodesk.Revit.DB.LinearArray	Class	Creates a linear array
Custom classes and methods	**Type**	**Description**
OffsetLines	Method	Creates an array reference line
LoadFamily	Method	Loads a family into the project
CreateAnArray	Method	Creates an array along the reference line

5.5.2.1 OffsetLines method

The offset command is included in the Revit user interface, but it is not implemented in the RevitAPI class, so I create the OffsetLines method using a series of default functions of the RevitAPI class. The method performs three tasks; creating two lines parallel to the Block family, checking whether the lines are inside or outside of the curve, and deleting the lines outside of the curve.

Before creating reference lines, the application reads edge information of the Block family objects through following steps. The solid of the Block family, faces of the solid, edges of the faces, and points of the edges are analyzed for all Block family models.

- Filter the Block family from the entire model.
- Access the GeometryElement class of the Block family, which is a geometric

representation of an element.

- Access the Solid class of the GeometryElement that has 3d solid information.
- Read all faces of the solid and select the top face where the array will be placed.
- Access the EdgeArray class of the faces that stores all face edges.
- Read line and point information of the edge.

Based on the edge information, the OffsetLines method creates two lines parallel to the filtered edge, check the line location, and then delete the line outside of the curve. To create two lines, line direction and vector values of the edge are calculated. Then by using the vector and the distance values, two lines are created both sides of the edge.

Between two lines, the line on the Block family surface is used for an array reference line. To filter the line outside of the Block family, the Project method of the Face class is used. The Project method can return distance from a projected point to a face, a nearest point on the face from the projected point, and a nearest edge object of the face from the projected point. It can also check whether the projected point is outside of the face as it returns a null reference.

Throughout the above process, the application generates a set of reference lines that will be used for tree array generation.

5.5.2.2 LoadFamily method

The LoadFamily method reads the plant type from the user interface and loads the corresponding plant family into the project. The method checks whether the family is already loaded for error handling. Based on the plant type, the method defines the family name, category, and symbol name. For example, when “American Beech – 20’” is selected from the user interface, the LoadFamily method defines the family category as “Planting”, the family name as “RPC Tree –

Fall", and the family file as "RPC Tree – Fall.rfa." To load a family symbol into the project through the API, all above values are required. When one family file, category, and symbol is switched to another, the LoadFamily method can place another families and the Array Generator can be utilized to another arrays such as lighting fixtures, street furniture, etc.

5.5.2.3 CreateAnArray method

After running the LoadFamily method, the CreateAnArray method assigns the array position, places the family symbol, and generates an array. The method accesses all reference lines, reads start and end points of the line, reads the tree spacing value from the user interface, calculates the required number of array members, places an array on the reference line. Currently, the LinearArray class of the RevitAPI is applied, so a linear array can be created by the application.

This method was applied for furniture, plant, and lighting fixture array generation. In array generation, following parameters (Table 12) need to be acquired. The element id is a parameter that decides the type of array members. The translationToAnchorMember indicates the vector value of the array. The anchorMember indicates rotation or translation value between the vector value and the array vector.

Table 12. Parameters of the LinearArray.Create method

Parameters	Type	Description
aDoc	Autodesk.Revit.DB.Document	The document
dBView	Autodesk.Revit.DB.View	The view that has the translation vector
id	Autodesk.Revit.DB.ElementId	The element id to array
count	System.Int32	The number of array members
translationToAnchorMember	Autodesk.Revit.DB.XYZ	The translation vector for the array
anchorMember	Autodesk.Revit.DB.ArrayAnchorMember	The rotation value that specifies the location of the second or the last array member.

5.5.3 Potential and limitations

Creating arrays through API follows the operation using commands Revit UI provides; selecting an array object (family) type, placing a start and an end point of an array, and selecting the number of array members. Placing an array position requires complex processes in the RevitAPI programming. Further details can be found in the offsetLines function of the PlantGenerator class.

5.6 UpdateArray for Editing the PUDM

UpdateArray is an application to update the type of arrays. When designers update multiple arrays in the PUDM, the UpdateArray can access only relevant arrays and update them automatically. For instance, if plant arrays having Doogwood trees need to update to Red Ash, the UpdateArray selects all Doogwood tree arrays from the entire project and then changes the tree type to the Red Ash for multiple arrays.

The UpdateArray can be used for various array types such as street trees, lighting fixtures, and street furniture. In this chapter, the plant UpdateArray is explained.

5.6.1 Description

The system operators can change the type of array members for one array by using existing commands in the Revit UI. Then they can update the last of arrays by running the UpdateArray. No custom user interfaces are provided for the UpdateArray, but following steps demonstrates how to use Revit commands and run the application to update all arrays.

First, select the first member of the array you want to update (Figure 24). The first or the last member of an array can be selected. Then click the Edit Group icon in the Group panel. After clicking the icon, the selected array will be deactivated. Select the same member you choose in

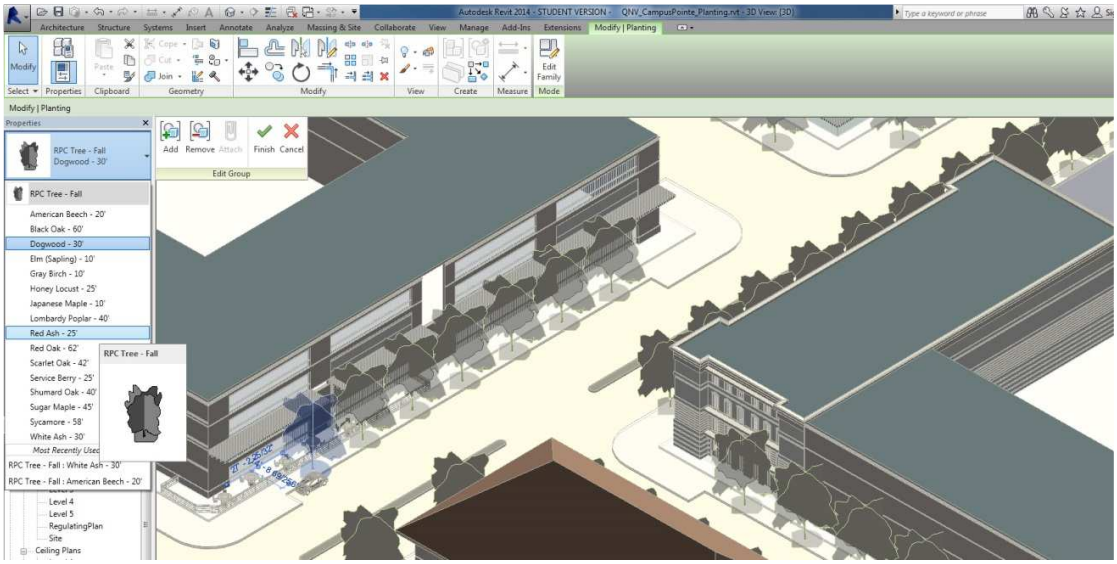


Figure 24. Choose a new tree type

the first step again. The array will be activated. At the *Properties* pane in the Revit UI, you can see the selected tree type. Choose the new tree type from the menu. The example demonstrates tree type changes from Dogwood - 30’ to Red Ash – 25’.

Click the Finish icon in the Edit Group panel. It will update all array members within the selected array (Figure 25). If the distance between trees needs to be updated, you can find and edit the number of array members near the center of the array. Right after you change the number, the array will be updated.

Final step is to update the last of arrays by running the UpdateArray (Figure 26). Go to the Add-Ins tab and click the Edit icon. The application filters the arrays having the same tree type and updates them automatically. For instance, when the tree array having Doogwood-30’ is updated, the application updates tree arrays having the same tree type only. If the number of the array members is updated, the application calculates the distance between trees in the selected array and then applies the same distance value to other arrays.

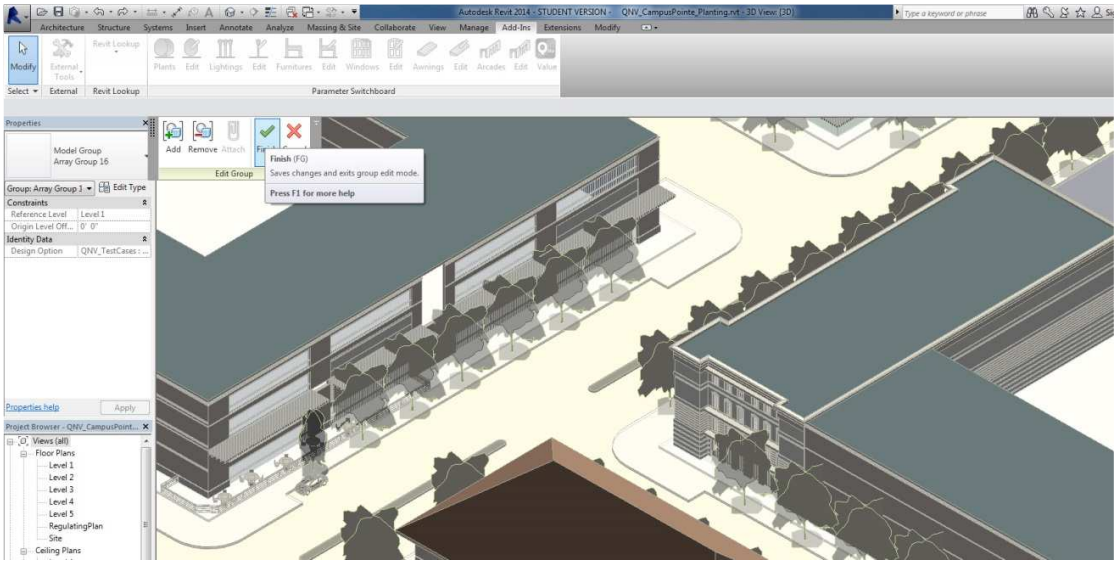


Figure 25. Click the finish icon to change other trees in the array

Figure 26. Run the Array type updater application

5.6.2 Application structure

The UpdateArray is an application that updates (1) the type of array members and (2) the spacing between the array members. After the system operators selects and edits an array by using Revit

commands, the application starts array updates based upon the user selection. From the selection, this application reads the type of array members and the spacing between the members. Next it filters arrays that have the same type with the selection. Finally, this application updates the filtered arrays to match them with the selection. To do so, the software development follows following steps:

(1) Initializing document

(2) *Reading the selected array information*: collects the array information. In tree array editing, for instance, the tree type, the length of the array, and the number of trees of the array are major array information.

(3) *Filtering arrays to be updated*: groups arrays that have same type of members, which will be updated by the application. If the tree array is selected, lighting or furniture arrays are excluded.

(4) *Updating filtered arrays*: applies the selected array information to the filtered array group. Depending on the array length, the application calculates the required number of array members.

(5) *Cleaning up*: deletes model lines that the application creates as array reference lines.

Revit stores array information in multiple locations. For instance, the tree type is stored in the Revit Family. The number of array members is stored in the Revit Model Group. The array location information can be found in the array object. To collect such dispersed array information, custom classes and functions are created in the application development.

5.6.3 Potential and limitations

Once an array is created in Revit, individual array members are grouped together. They belong to the Revit model group (OST_IOSModelGroups) category that is different from the Revit

Array category (OST_IOSArrays). When the application reads the selection, properties of the model group can be obtained as shown in Figure 27. To create, edit, and delete arrays, the application needs to retrieve both the array and the model group elements.

Some values such as the number of array members, the length of the array, the type of array members are not writable but readable from the application. For instance, the number of array members can be obtained from the model group (Members), but the value cannot be modified via the application. The selected array object in the Revit UI belongs to the Revit model group. The tree type cannot be accessible from the Revit model group, so the tree types cannot be edited from the selection. The Figure 27 shows attributes of the selected array that is an array group. There is no attributes for tree types. To resolve these limitations, the application filters an array and pairs it with the selected model group by using classes and methods of the RevitAPI class (Table 13).

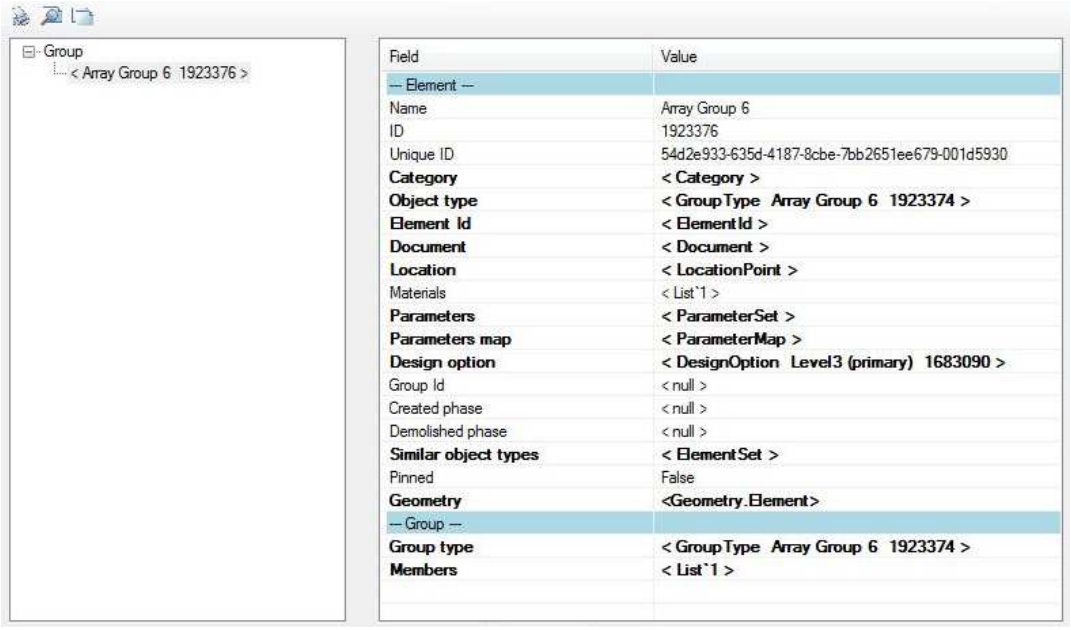


Figure 27. The properties of the Revit model group

Table 13. Major classes and methods of the UpdateArray application

Classes and methods of the RevitAPI	Type	Description
Autodesk.Revit.DB.Group.GetMemberIds	Method	Retrieves the id of the group that the array belongs
LinearArray.GetOriginalMemberIds	Method	Retrieves the original member Ids of the array
LinearArray.GetCopiedMemberIds	Method	Retrieves the copied member Ids of the array
Autodesk.Revit.DB.LocationPoint	Class	Retrieves the location of an element
XYZ.DistanceTo	Method	Returns the distance from the point to the specified point
LinearArray.NumMembers	Properties	Retrieves the number of the array members
Custom classes and methods	**Type**	**Description**
CalculateArraySpacing	Method	Measures the distance between array members
FilterArrayForUpdate	Method	Collects arrays to be updated
UpdateFilteredArrays	Method	Updates the type and the number of array members

Total number of the array members is retrieved from the LinearArray.NumMembers method. The position of the array is obtained from the first and the last array members that are retrieved by using the GetOriginalMemberIds and the GetCopiedMemberIds methods. Then the array length is measured by the XYZ.DistanceTo method, the first member location, and the last member location. From the number and the length of the array, the application calculates the required spacing between array members.

Once such additional properties are collected, they are stored in the extended array object. The extended array object has six additional properties: array group name, array id, the Revit Family name of the array member, the Revit Family Symbol name of the array member, and the number of total array members.

5.7 AssignTransectCodes for Assigning Transect Codes

AssignTransectCodes is an application that reads the type of transect zones of the Parcel objects, updates the transect zones of the selected parcels, and colorizes the transect types in the PUDM. The application enables to monitor the transect codes of the entire project and to update the transect types from the user interface instead of updating individual parcels manually. The colorized site plan corresponds to the Regulating plan in the recent urban regulations.

5.7.1 Description

The AssignTransectCodes application provides one UI that lists the current transect code types and allows users to modify the types from the combo box (Figure 28). Once new types are selected from the combo box, the “Code Update” button needs to be clicked to update the value.

5.7.2 Application structure

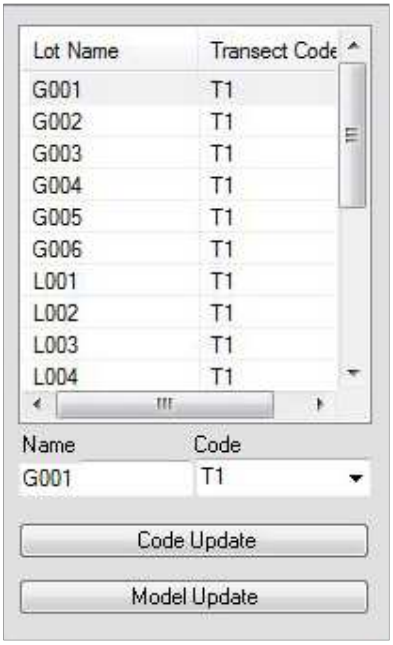


Figure 28. The user interface of the AssignTransectCodes application

The AssignTransectCodes application reads transect code information from the model, opens the user interface, and updates the transect codes according to the user inputs through the following steps.

(1) Initializing the document is the process that defines the Revit model and creates custom filters to locate related Code objects.
(2) Reading transect codes is the process that accesses the Parcel objects from the PUDM and reads the type of the transect codes.
(3) Opening the Transect Code Viewer is the process that sends the collected transect code types to the viewer and opens the viewer in the Revit interface.
(4) Updating transect codes is the process that receives the user inputs from the viewer, locates the related Code objects, and updates the transect code types.

Following custom methods are created in the AssignTransectCodes application (Table 14).

5.7.3 Potential and limitations

To visualize the transect codes with colors, the PURMs use both the Revit project parameters and the Revit filters. The visualization process is as follows:

(1) The application updates the Revit project parameter of the BIM object. The Revit project parameter needs to be predefined in the Revit project parameter menu.

Table 14. Major classes and methods of the AssignTransectCodes application

Custom classes and methods	Type	Description
AssignTransectCodes.ReadTransectCodes	Method	Reads the transect codes of the entire project.
AssignTransectCodes.UpdateTransectCodes	Method	Updates the transect codes and visualizes the transect codes with colors.

(2) According to the project parameter, the different types of the Revit filters are applied to the BIM object. The Revit filters needs to be predefined from the visibility/graphic override menu. Which Revit filter is applied for a certain project parameter is defined in the visibility/graphic override menu.

(3) Once a Revit filter is assigned, the surface color of the Code objects is applied according to the Revit filter.

In the PUDM, “CodeColor”, the project parameter of the Parcel objects are created in the Revit interface. In the “visibility/graphic override” menu, a series of the Revit filters are created. The type of the Revit filter is determined by the Revit project parameter, “CodeColor”, and each Revit filter determines the surface color of the Code objects. In sum, once the project parameter can determine the Revit filter and then the Revit filter can determine the surface color of the Code objects.

On the other hand, once the application changes the project parameter, the color of the Code objects can be updated. In the AssignTransectCodes application, the UpdateTransectCodes method updates the “CodeColor” parameter, the project parameter of the Parcel objects. Even though the process is explained with the AssignTransectCodes application, the same modeling and programming methods are applied to other applications that colorize the Code objects such as CheckCollision, VisualizeDensity, and ReportProForma.

5.8 AssignBuildingCodes for Testing Parameters

AssignBuildingCodes is an application that reads regulatory parameters of the Building objects, enables users to modify the selected parameters as required, and updates the entire Building objects. The AssignBuildingCodes application enables fast update of the Code objects when a set of parameters within the multiple Code objects needs to be tested.

5.8.1 Description

To run the AssignBuildingCodes application, it is required to select a Building object that will be updated. Then the application reads the regulatory information form the selection and presents the values in the user interface (Figure 29). To update the values, it is needed to select the parameters from the list box and then the selected value can be updated from the text boxes of the “Update Parameter” section. For each update, users should click the “Change Code” button to save the updated value. After all values are updated, clicking the “Model Update” button executes the AssignBuildingCodes application and the PUDM completes update.

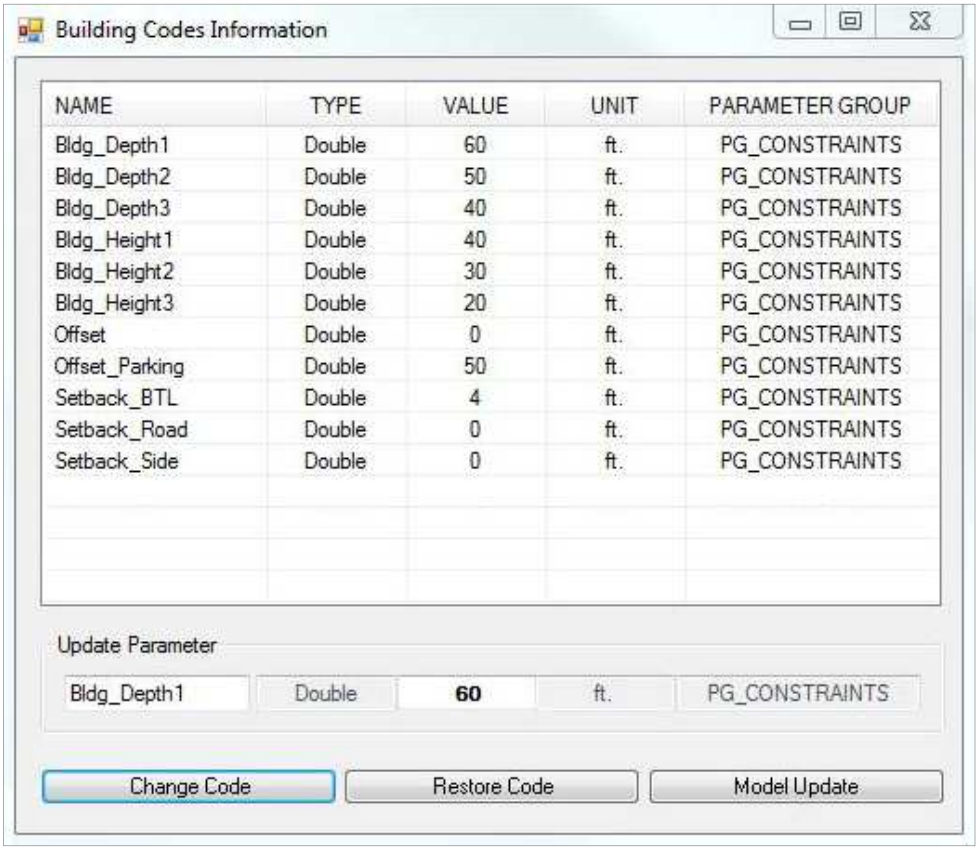


Figure 29. A user interface of the AssignBuildingCodes application

Although one Building object is selected, all Building objects belong to the same transect zones will be automatically updated by the application. This application, therefore, enables to test a set of parameters for the Building codes according to the transect codes.

5.8.2 Application structure

Reading and assigning parameter values are main methods in this application, which are often used in other applications. To do so, the application performs the following processes.

(1) Initializing the document is the process to define the Revit model, the custom filters, and the internal variables. A filter is created to locate the Building objects.
(2) Reading information of the selected Code object is the process to store parameter values from the selection.
(3) Opening the user interface is the process to show the parameter values of the selected code objects in the user interface. Updating parameters are performed in the user interface as I described.
(4) Collecting Code objects is the process to locate Code objects that belong to the same transect zones with the selection.
(5) Updating Building Codes is the process to update parameters of the Building objects according to the new values from the user interface.

5.8.3 Potential and limitations

While the application is created for the Building objects, the structure of the application is applicable to other types of the Code objects. If other Code objects such as the Block objects or the Parcel objects are selected and the corresponding custom filters are defined in the same structure, the application can perform the same or the similar tasks. Therefore, the very similar code structure is applied to the AssignParkingCodes application.

5.9 AssignParkingCodes for Editing Regulatory Variables

AssignParkingCodes is an application that reads regulatory parameters of the Parking objects, enables users to modify the parameters as required, and updates the entire Parking objects. The Parking objects that belong to the same transect zones with the selection can be updated simultaneously by this application.

5.9.1 Description

As shown in the AssignBuildingCodes application, the first step is to select a Parking object that needs to be updated. Then the user interface will be opened in the Revit interface (Figure 30).

The current form allows updating four parameters of parking regulations. One difference with the AssignBuildingCodes application is that the four parameters are stored in the varying Code objects; Open Space Ratio in the Parcel object, Require Parking Space in the Building object,

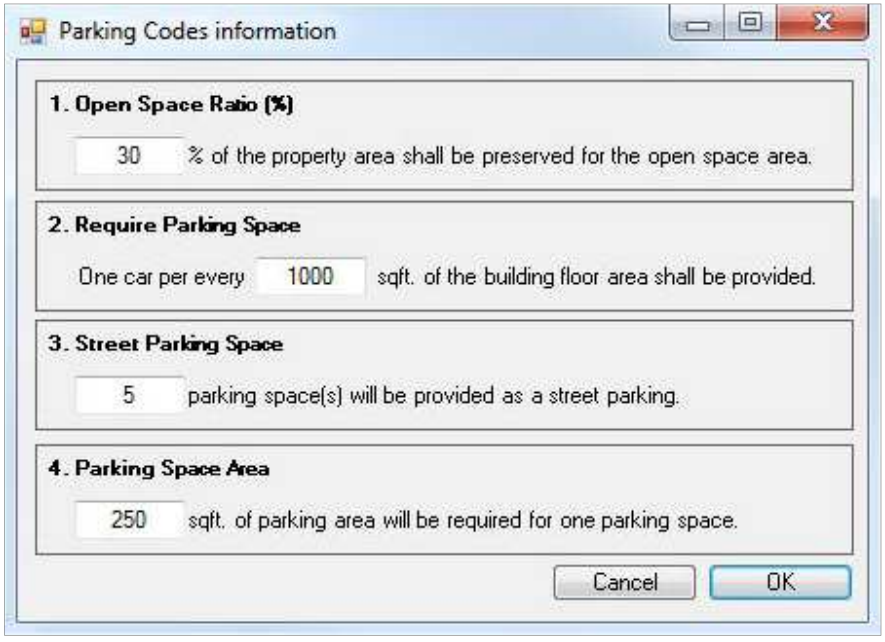


Figure 30. A user interface of the AssignParkingCodes application

Street Parking Space in the Parcel object, and Parking Space Area in the Parking object. On the other hand, all parameters in the AssignBuildingCodes application are captured only from the Building objects.

This application can read and update parameters, but no changes in the PUDM are occurred by the application. To update the Parking objects according to the new values, it is required to run the UpdateParking application.

5.9.2 Application structure

A similar process with the AssignBuildingCodes application is performed, but reading and updating parameters are more complex in this application.

(1) *Initializing the document*: defines the Revit model, the custom filters, and the internal variables. A set of the custom filters are created to the Parcel, the Building, and the Parking objects.

(2) *Reading information of the selected Code object*: identifies the related Code objects to the selection. The object hierarchy and nesting information among Code objects are used to locate all relevant Code objects to the selection.

(3) *Reading information of the relevant Code objects*: assesses the related Code objects and to read the four values from the respective locations.

(4) *Opening the user interface*: displays the parameter values in the user interfaces. A user can update the values in the user interface.

(5) *Collecting Code objects*: locates relevant Code objects that belong to the same transect zones with the selection.

(6) *Updating Parking Codes*: updates parameters according to the new values from the user interface. The Parcel, the Building, and the Parking objects are updated.

5.9.3 Potential and limitations

The application structure is very similar to the AssignBuildingCodes application. Both applications are intended to be used by code makers to test a set of code provisions prior to the code adoption. Even though applications for the open space standards and the streetscape standards have not been implemented, it may be possible to apply the very similar code structure.

Also, the parameter values shown in the user interface are assumed not only as writable for the code makers but also as readable for the code users. It implies that the PURM may be shared by both the public sectors such as planners and code makers and the private sectors such as designers, engineers, and citizens in different purposes.

5.10 UpdateParking for Updating Models through Matrixes

UpdateParking is an application that (1) collects parking regulatory information from the Code objects, (2) calculates the required parking spaces according to the development capacity, and (3) updates the parameters and dimensions of the Parking objects.

The UpdateParking application updates the Parking objects according the common rules found in the urban regulations or in the practice. In the typical urban regulations, the capacity and the type of buildings determine the parking space number. Also, the area of the parking spaces is associated with the site area, the building footprint area, and the open space area within the site.

In the PUDM, such information is imbedded in the multiple Code objects, so the location of the individual values and their relationships needs to be specified in the application. Such common rules and the specifications are used in the application development for the parking space calculations.

5.10.1 Description

The UpdateParking application does not provide user interfaces, but the interface of the AssignParkingCodes enables users to modify the regulatory parameters for the parking plan.

5.10.2 Application structure

The first step is selecting a parcel model to read its name, which will be used for locating a correspondent building model and a parking model from the entire PUDM (Figure 31). To do so, I retrieve the name value from the Parameters class of the parcel model. After that, other internal variables of the model name and the parcel name are defined. A loop construct of foreach is used, so the same process can be applied for entire parcels.

Next step is collecting required parametric values for the density calculation from the selections. The property area, transect zoning codes, and the parcel name are collected and the data type of them is converted for the following process.

This is an entry point to access a building model. I named the building model as the codeModel. The useful parameters are stored in the Parameters class of the codeModel, so I define an internal parameter set with this class. The area of total building floors can be obtained from the building model parameters. The parameter name value is “Gross Floor Area”, and the value is stored as a double type. I store the value into the internal variable of TFA with an integer. Using

```
if (faNames.Contains(lotName))
{
    totalFloorArea
     = codeModel.get_Parameter("Gross Floor Area").AsDouble();
    int TFA = (int)totalFloorArea;

    /// Calculate Floor Area Ratio
    floorAreaRatio = totalFloorArea / parcelArea;
    string FAR = floorAreaRatio.ToString().Substring(0, 4);
```

Figure 31. A code block for calculating density using internal variables

two variables of the parcel area and the total floor area, I calculate F.A.R. that can show the development density. The last step is to store internal variables into a parameter list as a comma separated string, which will be presented in the viewer.

5.10.3 Potential and limitations

In this research, I implemented the parametric urban design code model using BIM and OOP. It can visualize key provisions of urban design codes and the structure of code components. The API in BIM can facilitate interactive communications among multiple code objects, which can assess the information within the BIM objects and can generate values from model parameters. Code provisions governing quantitative aspects of urban design, such as urban form and capacity can be modelled. In addition, associations across code components and their provisions also can be established in BIM objects using customized applications. The findings show the potential of the parametric modelling and API in BIM for the real-time design evaluations and model changes during the urban design process.

5.11 CheckCollision for Checking Code Compliances

CheckCollision is an application that detects the Code object intersected with others. For instance, when a Building object is intersected with a Parking object, this application can detect them, generate reports, and colorize both objects in the PUDM.

5.11.1 Description

The CheckCollision application provides two user interfaces. One is a notification window (Figure 32) and the other is a result viewer (Figure 33).

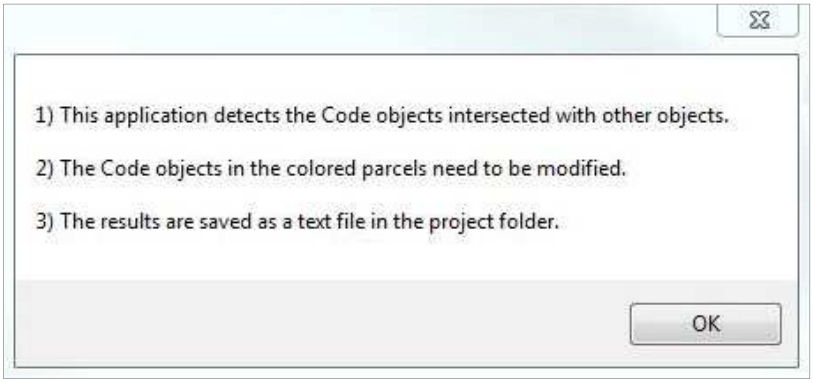


Figure 32. A notification window of the CheckCollision application

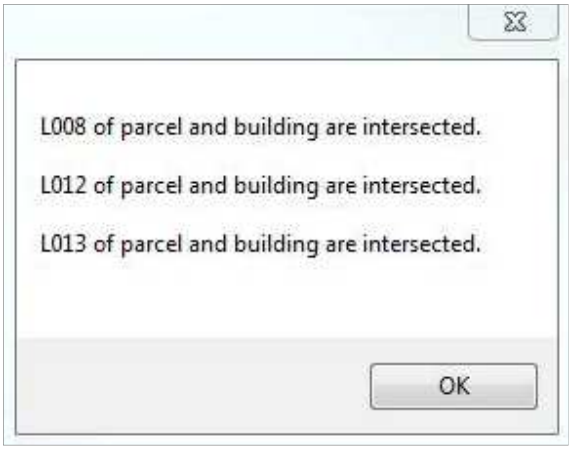


Figure 33. A result viewer of the CheckCollision application

After the application is closed, the intersected Code objects are colorized in the PUDM. After the Code objects are transformed by parameter changes, this application enables operators to identify the problematic parcels to be fixed. The excessively large Building and Parking objects compare to the Parcel objects can be located with this application.

5.11.2 Application structure

Complex operations to analyze geometry information of the Code objects are performed in the

CheckCompliance application through the following steps.

(1) *Initializing the document process*: creates custom filters to locate the Code objects.

(2) *Reading geometry information*: collects geometry information of points, lines, and surfaces of the Code objects. The Block, the Parcel, the Building, and the Parking objects are analyzed and the collected information is grouped by parcels.

(3) *Checking intersection*: finds intersected segments among the grouped geometry information in the step 2.

(4) *Coloring the intersected Code objects*: colorizes the intersected Code objects.

(5) *Reporting the results*: generates a report file.

Table 15 shows custom methods of CheckCollison.

5.11.2.1 CheckIntersect method

To check all the intersected objects, two objects are compared at once. E.g., A Block object and a Building object or a Building object and a Parking object. Geometry information of the two

Table 15. Major classes and methods of the CheckCompliance application

Classes and methods of the RevitAPI	Type	Description
IntersectionResult	Class	Gives the intersecting geometry entities
Line.Intersect	Method	Calculates the intersection of the line with others
Face.Project	Method	Projects the specified point on the face
Custom classes and methods	**Type**	**Description**
CheckCompliance.collectGeometry	Method	Reads geometry information of the Code objects.
CheckCompliance.checkIntersect	Method	Detects intersection among the Code objects.
CheckCompliance.reportResults	Method	Generates result reports

objects is stored into two groups and the checkIntersect method compares one to another. This method utilizes the IntersectionResult class in the RevitAPI. The IntersectionResult class can return the distance between two objects, the closest edge of one object from another object, and the intersection point coordination. When the intersection points exist and the points are on the objects' surface, the checkIntersect method determines that the Code objects are intersected each other.

5.11.3 Potential and limitations

The application detects intersecting objects, but minor changes in its methods may enable to perform various code compliance checking. For instance, by using the similar classes and methods of this application, the setback incentive regulation can be checked. If the setback allowances according to building height are parameterized as a BIM object such as lines, surfaces, or masses, it is possible to check whether the setback object in BIM is intersected with other Code objects. In addition, the minimum distance between two objects can be easily measured by using the same methods, the Face.Project method, used in this application. The provisions limiting a maximum or a minimum distance between two objects may be represented with the very similar approach found either in the Line.Intersect method or in the checkIntersect method.

5.12 VisualizeDensity for Visualizing Performances

VisualizeDensity is an application that (1) analyzes density of the project and the individual parcels, (2) displays the analysis results in the window, and (3) visualizes the distribution of density with colors in the PUDM.

To calculate density, two parameter values need to be collected: the lot area from the Parcel objects and the gross floor area from the Building objects. The application collects the lot area

and the gross floor area of individual lots, calculates the gross floor area of the entire project, and calculates average density. Floor Area Ratio (FAR) is used to present density.

This application enables the fast analysis of density upon continuing updates of the PUDM. The density distribution of the individual sites can be colorized in the PUDM, so that users can identify which sites have higher or lower density under certain parameter settings.

5.12.1 Description

The VisualizeDensity application provides a display window showing analyzed model information in the project and the parcel level (Figure 40). Also the visualization setting controller in the bottom enables a wide range of colors to be visualized in the PUDM. As larger values are selected in the “Color Scale” bar, the more colors are applied to show the detailed density distribution.

5.12.2 Application structure

The VisualizeDensity application reads model information, calculates density of the parcel and the project level, displays a list of values, and visualizes density in the PUDM via the following processes.

(1) *Initializing the document*: defines the Revit model and filters.
(2) *Reading model information*: reads the site area of the Parcel objects and the gross floor area of the Building objects. Then stores the values.
(3) *Calculating density*: by using a simple equation, calculates density of individual lots. The overall floor area and the site area of the entire project are also calculated.
(4) *Displaying the values*: opens the density viewer and displays the collected values. Inquires the visualization setting values using the controller.

(5) *Visualizing density*: colorizes individual Parcel objects. The deviation to average density is visualized with a series of colors.

The classes and the methods in this application are created in the very similar approach of other applications. For instance, reading model information is similar to the functionality of the AssignBuildingCodes application. Calculating density is performed through a simple equation using the parcel area and the gross floor area. The density viewer is similar to the interfaces of the AssignTransectCodes, the AssignBuildingCodes, and AssignParkingCodes applications. Visualization of density is performed as the CheckCollision application does.

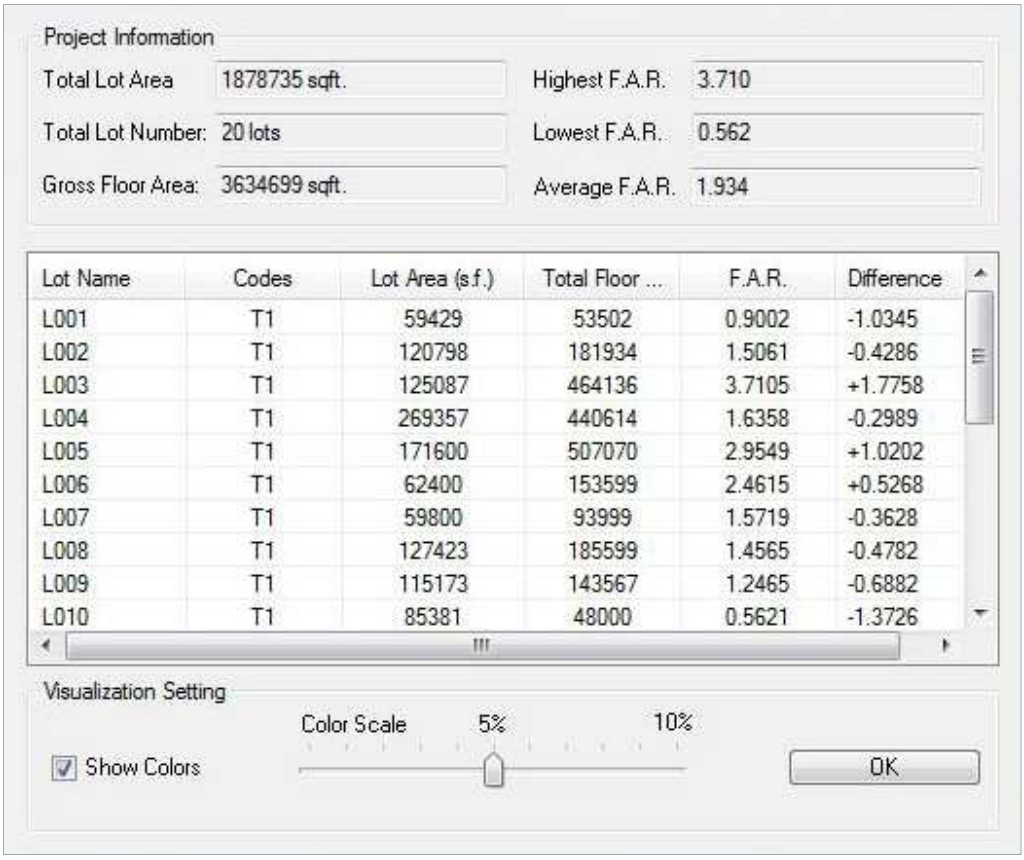


Figure 34. A user interface of the VisualizeDensity application

5.12.3 Potential and limitations

While density is visualized in this application, other information can be expressed with colors. The AssignTransectCodes application visualizes the transect codes with colors. The CheckCollision application colorizes the intersected Code objects. This visualization approach may be utilized to present other object-level analysis results with objects' colors, enabling stakeholders to understand the distribution of the analyzed values.

5.13 ReportProForma for Analyzing Performances

ReportProForma carries out an economic analysis for multiple design scenarios in a single run. The application is developed based on the simplified Pro Forma analysis methods, custom classes and methods, as well as the features of the PUDM.

The ReportProForma application carries out the following processes:

(1) Applies the first parameter set to the PUDM.
(2) Refreshes the model view in the Revit UI to show the updated model.
(3) Analyzes Pro Forma and stores the analysis results with the applied parameters in the spreadsheet.
(4) Repeats (1) ~ (3) for the last of parameter sets.
(5) After applying all parameter sets, the Pro Forma reports and the model information reports are opened.

In the application, the following processes are implemented.

- *Updating parameters*: changes the model parameters to update the PUDM. A various combination of parameter values can be applied to the model parameters.
- *Performing an economic analysis*: estimates project costs and profits based on the

simplified Pro Forma analysis method.

- *Generating reports*: creates and opens a model information report and a Pro Forma report.

5.13.1 The simplified Pro Forma analysis method

The ReportProForma application is implemented by using the simplified Pro Forma analysis method to perform BIM-based Pro Forma estimation with the PUDM. The simplified Pro Forma analysis method is developed based on the Pro Forma calculation method of Gerald D. Hines, a real estate development company. The analysis process has five criteria as shown in Table 16: project information, total project cost, net operating income (NOI), gross revenue, and development profit.

The analysis requires external data such as the unit cost of land, design, and construction as well as development information such as gross floor area and uses of the buildings. Once external data and development information are collected, the other values in the table are calculated by using equations of the application. In the application development, an external data is stored as parameters, development information is retrieved from the PUDM, and calculated values are stored in the parameters in BIM.

5.13.2 Description

The ReportProForma application runs without user interfaces. As a user runs it, the application updates the parameters of the Code objects, conducts the Pro Forma analysis, and generates a model information report and a Pro Forma analysis report. Two reports can be used to identify how parameter changes affect the financial feasibility.

Table 16. A simplified Pro Forma analysis in the ReportProForma application

Project Information			
1	Site Area	sqft.	From BIM models
2	Gross Floor Area	sqft.	Collect gross floor area per building use
Total Project Cost			
3	Site Cost	$	Unit cost of land ($ per sqft) x site area
4	Architect & Engineer	$	Gross Floor Area (2) x Unit cost (4)
5	Construction	$	Gross Floor Area (2) x Unit cost (5)
6	Tenant Improvements	$	Gross Floor Area (2) x Unit cost (6)
7	Taxes/ Insurance/ Permits/ Fees	$	Gross Floor Area (2) x Unit cost (7)
8	Financing	$	Gross Floor Area (2) x Unit cost (8)
9	Leasing/Sale Costs	$	Gross Floor Area (2) x Unit cost (9)
10	Public Relations/ Marketing	$	Gross Floor Area (2) x Unit cost (10)
11	General and administrative expenses	$	Gross Floor Area (2) x Unit cost (11)
12	Contingency	$	Gross Floor Area (2) x Unit cost (12)
13	Total Investment or Project Cost	$	
Net Operating Income			
14	Net Lettable Area	sqft.	Gross Floor Area (2) x Efficiency (15)
15	Gross $ rental rate of leasable space	$ per sqft.	Rental rate per building use
16	Occupancy Rate (1- Vacancy Rate)	%	Occupancy per building use
17	Gross Revenue	$	Net lettable area (15) x rent (16) x occupancy (17)
Gross Revenue			
18	Efficiency Rate of Lettable Space	%	Exclude public area such as corridor, elevator, etc.
19	Net Lettable Area	sqft.	Gross Floor Area (2) x Efficiency (18)
20	Gross Rental Rate per sqft. of Leasable Space	%	
21	Occupancy Rate	%	
22	Gross Revenue	$	Net lettable area (19) x Rent (20) x Occupancy (21)

Table 16. Continued.

23	Operating Expenses of Leasable Space	$ per sqft.	
24	Net $ rental rate of Leasable Space	$	Rent (20) - Operating expenses (23)
Capitalized Value and Development Profit			
25	Gross Revenue	$	Gross revenue (22)
26	Net Operating Income (NOI)	$	NOI (16)
27	Capitalization Rate	%	
28	Sale Price	$	NOI (26) / Capitalization Rate (27)
29	Project Cost	$	
30	Development Profit	$	Sale price (28) - Project cost (13)

The application reads predefined data from the parameters, retrieves Revit model information, and calculates new values to create a Pro Forma report. For instance, the unit cost of land is predefined in the Site object. The site area is a model property stored in the Site object. Then, the total site cost is calculated by using the above two values.

The application tests a range of the parameter values such as building height, depth, and setbacks, which are currently defined within the application. An interface will be added to allow stakeholders to modify the type and the range of parameter values. The values for the Pro Forma analysis such as the unit costs, efficiency, and occupancy are stored in the PUDM, so the system operator can edit the values in the Revit UI. For instance, by selecting the Site object in the Revit UI, I can see and edit the predefined parameters from the property window (Figure 35).

5.13.3 Application structure

The ReportProForma application can apply a series of parameter sets to the PUDM, perform the Pro Forma analysis, and generate the result report. The application uses the custom methods and

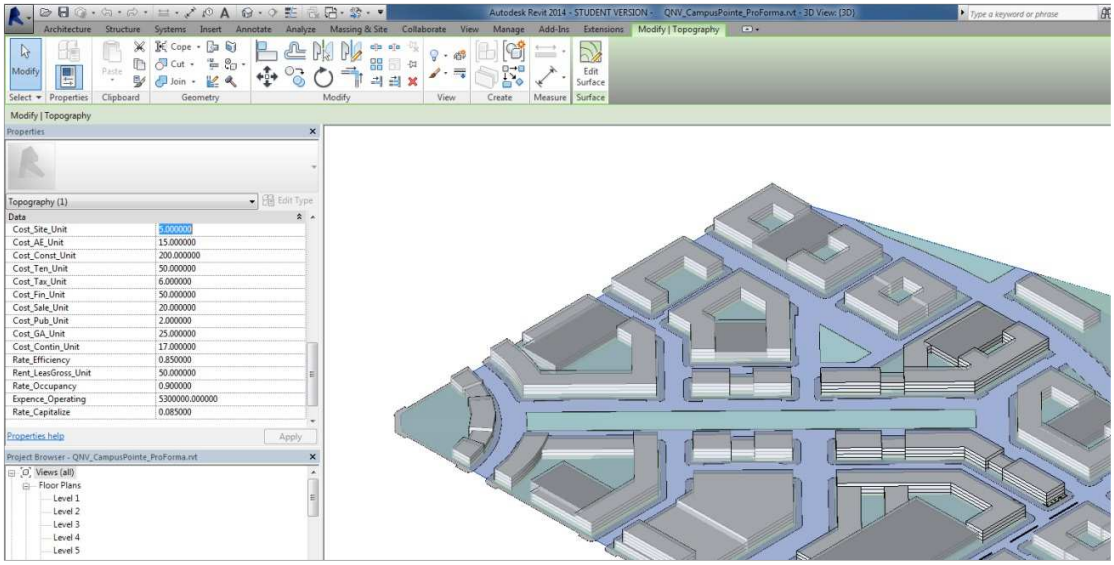


Figure 35. Parameter changes for the Pro Forma analysis

classes as well as calls the external custom applications to perform the multiple processes in a single execution. For users, this application is easy to use, but complex processes are carried out in software point of view.

The ReportProForma application uses (1) the UpdateParameter method to change model parameters, (2) the CreateModelInfoReport method to create a report of the model information, and (3) the GenerateProForma application that has the AnalyzeProForma method and the CreateProFormaReport method (Table 17).

The urban model is being updated while the UpdateParameter method runs. The model information report and the Pro Forma report are opened after the GenerateProForma runs.

5.13.3.1 UpdateParameter method

The PUDM consists of a series of the Code objects that are created using the Revit Families; Site, Block, Parcel, Building, and Parking. Each Code objects holds parameters either controlling

Table 17. Classes and methods of the ReportProForma application

Classes and methods of the RevitAPI	Type	Description
ClosedXML.Excel.XLWorkbook	Method	Creates a new Excel workbook
Custom classes and methods	**Type**	**Description**
UpdateParameter	Method	Updates the parameter sets
CreateModelInfoReport	Method	Creates a model information report
GenerateProForma	Class	Performs the Pro Forma analysis
GenerateProForma.AnalyzeProForma	Method	Analyzes Pro Forma of the PUDM
GenerateProForma.CreateProFormaReport	Method	Creates the Pro Forma report

the object geometry or storing supplementary information. The UpdateParameter method changes the parameters affecting the geometry such as building height, depth, and setbacks. For example, the Building object has a parameter of the number of floors. Changing the parameter value via the application affects the object height. Following classes and methods are used in the UpdateParameter method (Table 18).

The Revit Transaction class is an object that evokes any changes made to a Revit model. The current application runs 12 transactions, and one transaction updates all lots. The sequence of the parameter update can be adjusted according to the arrangement of multiple transactions.

Table 18. Major classes and methods of the PUDA

Classes and methods of the RevitAPI	Type	Description
Autodesk.Revit.DB.Transaction	Class	Guards any changes made to a Revit model
Element.get_Parameter	Method	Retrieves the element parameter
Parameter.Set	Method	Sets the new parameter

5.13.4 Potential and limitations

The ReportProForma application runs without user interfaces. If a user interface will be created to allow stakeholders to select the type of parameters and their ranges, more diverse scenarios of the urban regulations can be studied with the ReportProForma application.

The ReportProForam application shows how an economic analysis can be carried out with the custom applications in parametric BIM and the BIM objects. The required data stored in the BIM parameters can be retrieved by the application. The calculated values can be stored in the parameters of the BIM objects. The equations of the calculation algorithm are built in the application. To do so, the BIM objects should store required information as the parameters. The applications should collect the required values. The accurate equations need to be built in the application to calculate the new values.

5.14 Summary

This chapter illustrated the prototype development process. First, I explained common difficulties in the urban regulation modeling process and assigned 9 tasks to be addressed by the PUDAs. Then I created 11 prototypes and discussed the prototyping scope, the working process, technical issues, potentials, challenges, and limitations.

The GenerateArray and the UpdateArray applications support the modeling process. Most applications allow reading and writing regulatory variables in the PUDM. The AssignTransectCodes application enables editing the transect codes. The AssignBuildingCodes enables updating some regulatory variables of the Building Envelope Standards. The AssignParkingCodes and theUpdateParking applications update the Parking Objects according to the rules to determine estimate the required parking number, available lot area for parking, and the scale of the parking structure.

Three applications are created to demonstrate the analysis process in the PURMs. The VisualizeDensity application calculates the development density and visualizes the density distribution with the color codes in the PUDM. The CheckCollision application filters the intersected objects and visualizes the filtered objects with the color codes. The ReportProForma application performs a BIM based economic analysis and exports the analysis results into the spreadsheet.

In the next chapter, I will demonstrate how individual applications can work with the PUDM for the existing zoning regulation sample.

6 EXPERIMENT AND EVALUATION

In this chapter, I describe the experiments of PZRM with Form-based Code Station Area of Farmers Branch. The experiment is designed to provide empirical evidences of how a part of the urban regulation making process can be addressed by PURMs. First, I implement the PUDM and the PUDOs of the selected zoning codes by using the modeling rules and the process established in Chapter 4. To do so, I establish a modeling scenario of PURMs. Each phase in the scenario will be processed using a set of PUDAs created in Chapter 5. Three sub propositions in Chapter 1 will be tested throughout the experiments. In addition, I will describe the limitations and challenges of PURMs for existing zoning code implementation.

6.1 Experiment Design

6.1.1 Scenario

For the experiments, I establish an experimental scenario with a sequence of steps (Figure 36). In each step, I will demonstrate the urban regulation modeling and the analysis processes by using the PUDM and a series of PUDAs. The experimental scenario is as follows:

6.1.1.1 Step1: create PUDOs

This is the first step to create a PUDM. Individual PUDOs will be created according to the rules and the process investigated in Chapter 3.

6.1.1.2 Step2: create a PUDM

Then a PUDM will be assembled with the PUDOs. In assembling, PUDAs will be used.

6.1.1.3 Step 3: assign parameters

Parameters of individual PUDOs will be assigned by using PUDAs. A series of PUDAs will be used to assign the new parameter values imbedded in Block, Parcel, and Building objects.

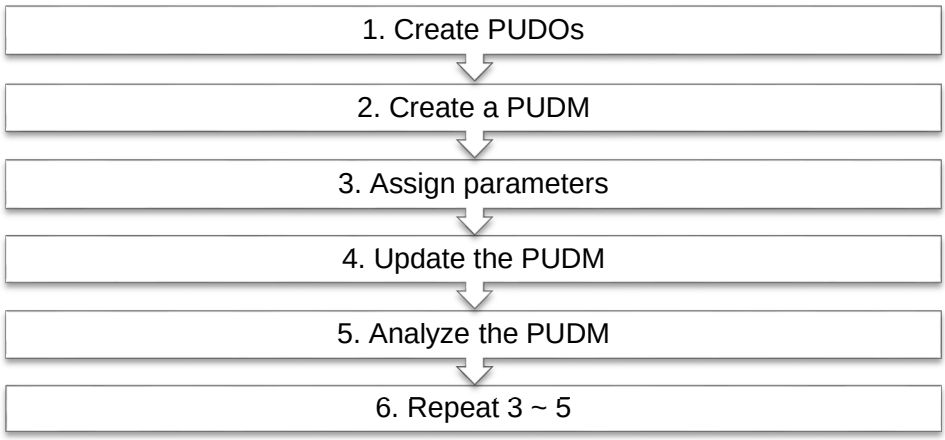


Figure 36. An experimental scenario

6.1.1.4 Step 4: update the PUDM

The parameters interrelated to other parameters will be updated in this step. Many parameters of the Parking objects will be updated according to the parameters of the Block, the Parcel, and the Building Objects assigned in the previous steps. To update the PUDM, a series of PUDAs will be used to read the parameter values, calculate the new parameter values, and assign the valued to the Parking Objects.

The parameters in this step pose are interrelated with other parameters, so they can be calculated by using the PUDAs. This section shows an example of how the urban regulation provisions having a relationship with other provisions can be represented in the PURMs.

In Chapter 4, I explained that the PUDOs have four types of parameters: spatial regulatory, non-spatial regulatory, performance, and supplementary. In step 2, spatial regulatory parameters are mainly assigned, while spatial regulatory parameters, non-spatial regulatory parameters, and supplementary parameters are assigned in step 3.

6.1.1.5 Step 5: analyze the PUDM

By using PUDAs, a series of analyses will be performed to demonstrate the analysis process in the PURMs. Exemplified development performances in this experiment are urban form, development density, economic footprints, and energy performances of the zoning code schemes. I assume that the step 2~4 can be repeated until the analysis results meet the objectives in the urban regulation modeling.

In this scenario, two types of operators are expected to use PURMs.

Parametric Urban Regulation Expert: a Parametric BIM expert in parametric modeling and BIM API application developments who can create the PUDOs, the PUDM, and the PUDAs. The step 1 and 2 are expected to be performed by the Parametric Urban Regulation Expert.

Parametric Urban Regulation Analyst: an urban planner or a zoning code creator involved in the zoning code making process. Prior to the zoning code adoption, the analyst will be able to test a range of provisions, analyze the zoning code performances, and predict the development performances. The Parametric Urban Regulation Analyst are expected to perform the step 3, 4, and 5.

In this research, I will conduct the experiments as an expert and an analyst of the PURMs.

6.1.2 Propositions of this experiment

In this experiment, three propositions will be addressed.

(1) The PURM based on BIM can represent common, typical, and important provisions of urban regulations. This will be addressed in the step 1 and 2 in the experimental scenario.
(2) The PURM based on BIM can enable tests of consequences of candidate zoning

provisions before adoption of a zoning ordinance. This will be addressed in the step 4.

(3) The PURM based on BIM can reduce ambiguity in interpretation during the design phase of a building project. This will be addressed across the experimental scenario.

The demonstration in the following sections will provide empirical evidences to address the three propositions.

6.2 Form-based Code Station Area of Farmers Branch

I create a PURM using an existing zoning code, Form-based Code Station Area of Farmers Branch. The zoning code is one of four selected code examples in Chapter 3. This zoning codes was adopted to regulate developments near the DART station located in north of Dallas. Since zoning code was adopted in 2005, the regulating plan and the provisions have been amended.

Figure 37 is the regulating plan amended in 2012. *Mustang Station* is a mixed use development adjacent to the DART station (A left image of Figure 38). The four story building consists of 257 residential units and about 10,000 square feet retail space on the ground level. *Mustang Crossing* is a residential development for 29 single family houses (A right image of Figure 42). Figure 39 shows a part of building envelope standards. *Mustang Station* is regulated by these standards such as building height, siting, façade treatment, and use.

The focusing area of this case study includes 10 lots near the station. These blocks are close to city's landscape corridor, the community parks, and the city hall. New projects are developing under this zoning code in this area.

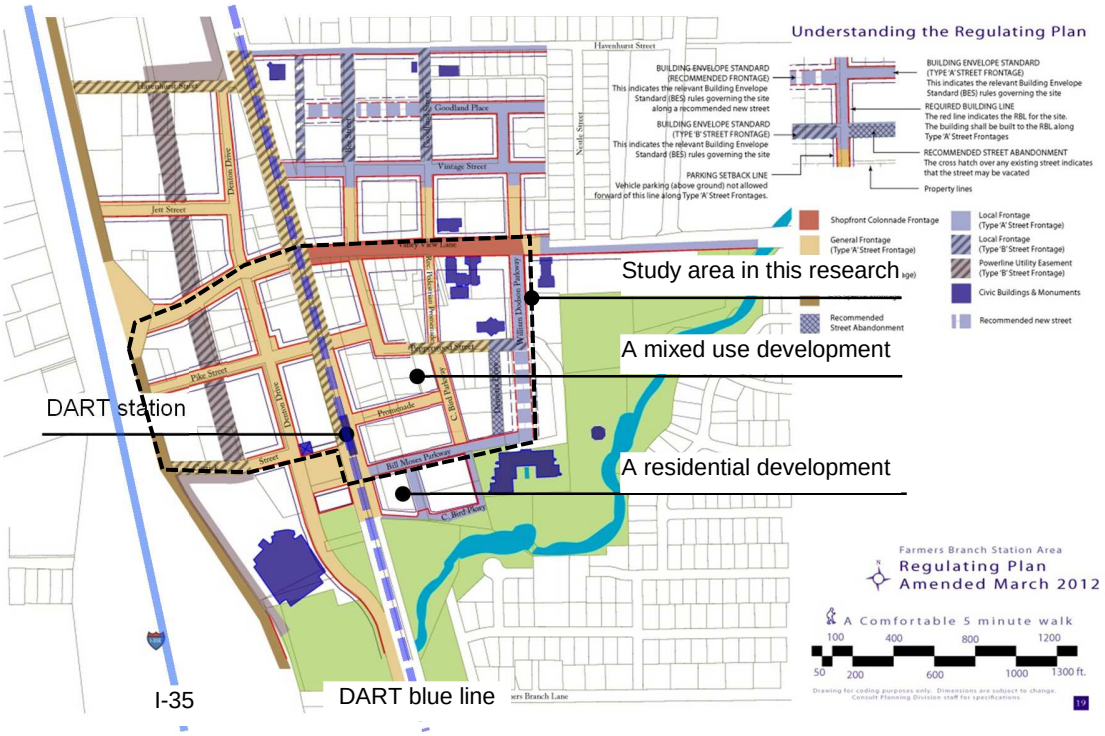


Figure 37. The regulating plan. Adapted from Form-based Code Station Area of Farmers Branch

Figure 38. A mixed use development (Left) and a single family residential development (Right) near the DART station (April 2014)

In the regulating plan, the color code on streets indicates the type of building frontage and the

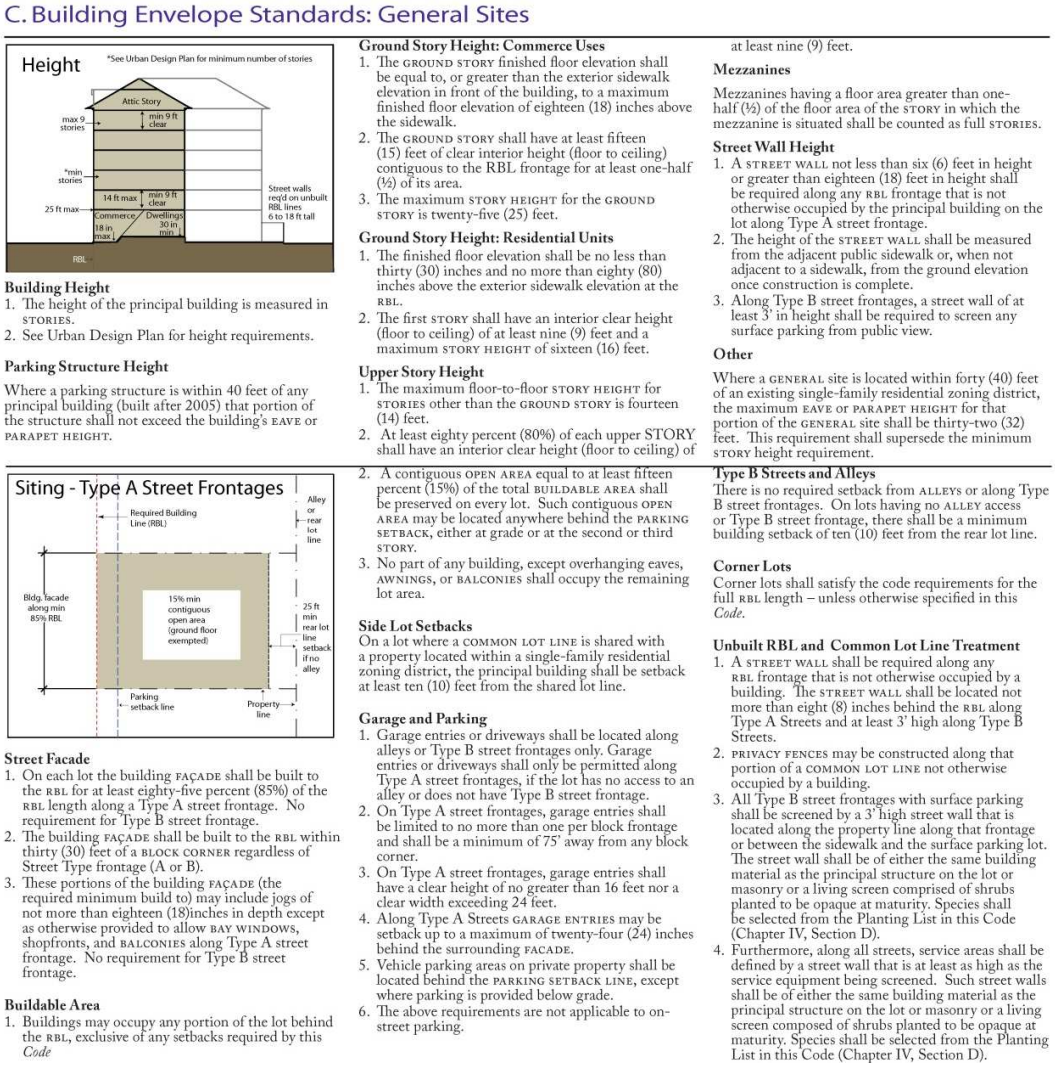


Figure 39. Height and siting regulations in Building Envelope Standards for the mixed use development. Adapted from Form-based Code Station Area of Farmers Branch.

building envelope standards. As we can see, most lots in the focusing area are surrounded by the yellow color roads. It means that the frontage type of the sites is *General Frontage* and the required information can be found in the building envelope standards for *General Frontage* (Figure 39). The urban design plan designates 1~3 story building height and seven street types for the study area.

6.3 PURMs for Farmers Branch

Based on the scenario, I created and tested PURMs for the Farmers Branch zoning codes. The detailed process of each step is explained in this section and observations and results will be discussed in the next section.

6.3.1 Create PUDOs

First, I created a set of PUDOs for 10 parcels based on the parametric modeling rules specified in Chapter 4. For each parcel, I built the Block, the Parcel, the Building, and the Parking objects. The Block objects are created based on the regulating plan and the streetscape standards. The Building objects represent form implication of the urban regulations such as the RBL and setbacks in the regulating plans and building height regulations in the building envelope standards. The Parcel objects are modeled using the site topology in the regulating plan and open space standards. The Parking objects are created based on "Rules for New Development" in the regulating plan. This zoning code does not provide a standalone parking standards.

6.3.2 Create a PUDM

A PUDM is a combination of a series of PUDOs. I created the individual objects manually and then assembled them automatically by using a PUDA and the default Revit functions.

By using the GeneratePlan application (Chapter 5.4), I assembled a series of PUDOs to create a PUDM. During assembling, one object is nested by the other object's surface so that an object hierarchy was established among Block, Parcel, Building, and Parking.

Then I added tree arrays on the Block object by using the GenerateArray (Chapter 5.5). As shown in Figure 40, a range of tree types and the positions can be selected in the user interface. After creating tree arrays, the tree types, tree spacing, and positions were edited by using the

UpdateArray application (Chapter 5.6).

6.3.3 Assign parameters

6.3.3.1 Assign the transect codes

As a PUDM is created, the first step is to assign the transect codes to the Parcel objects, indicating the type of building envelope standards. In Farmers Branch zoning codes, the type of streetscapes and building frontage types are defined in the regulating plans. In PURMs, the type is stored as a parameter of the Parcel object by using the AssignTransectCodes application. A user interface is opened to ask the type of transect codes, the application finds the Parcel objects, and updates the transect code parameters. As the parameter value is updated, the Parcel object color is changed in Revit (Figure 41). When a set of parameters needs to be applied to a specific transect zone, the transect code parameter enables filtering the relevant objects from the PUDM.

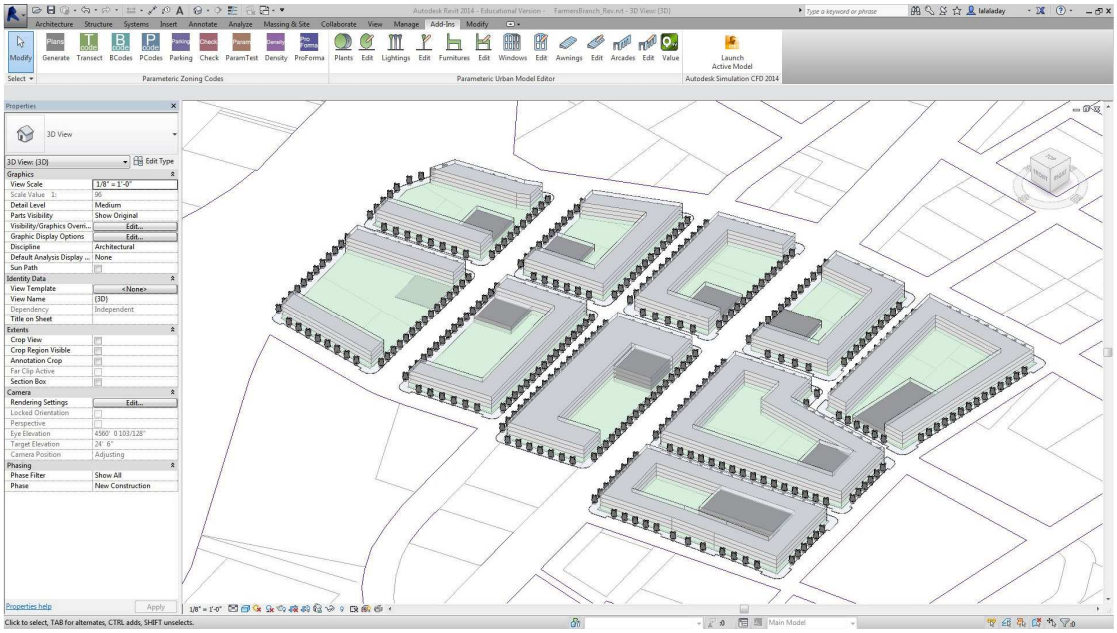


Figure 40. The PUDM of Farmers Branch

The approach of assigning transect zones in the regulating plan varies over the zoning codes. Often, the transect zone is assigned to the lots, which is similar to the typical land use map. In this case, the required regulation standards can be located according to the transect zones. The PURMs are built upon this approach. As shown in Table 2 of Chapter 3, the zoning codes of Fort Worth and Ventura assigned the transect codes to the lots. The PURMs is built upon this approach.

In Farmers Branch, the street types and the building frontage types are defined in the regulating plan without using the transect codes. Some parcels can have multiple building frontage types because the building frontage type is assigned on the street. In the regulating plan, the General Frontage type is dominantly assigned, so I applied this type to ten parcels.

6.3.3.2 Assign the building codes

The Building objects correspond to the building envelope standards. The parameters of Building

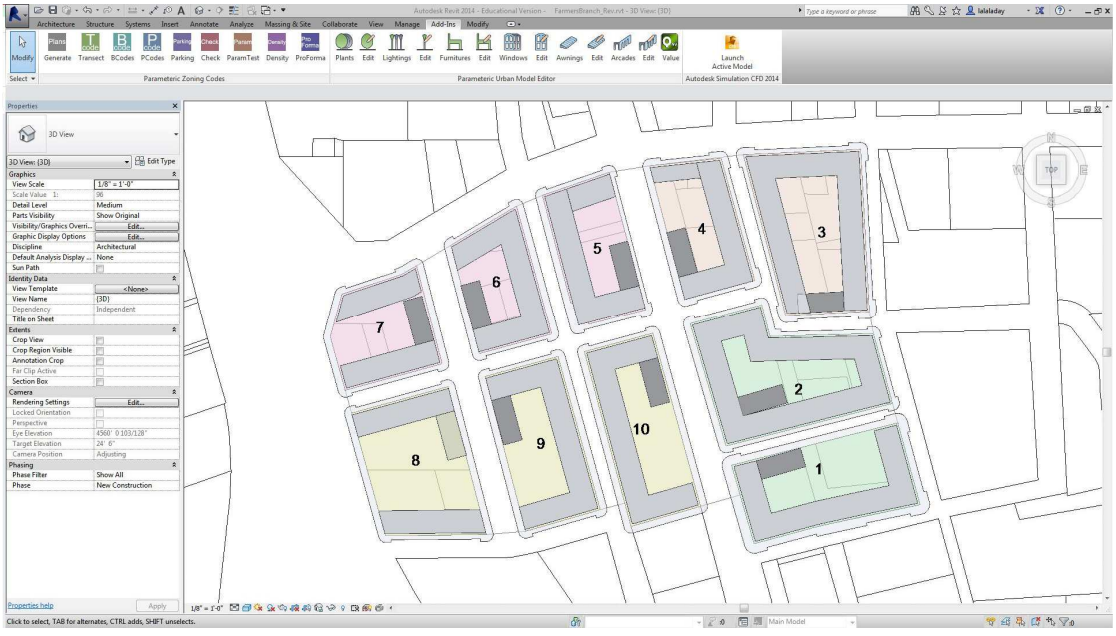


Figure 41. The transect codes are visualized with the color codes.

objects include building height, depth, setback, and use, which can be commonly found in the building envelope standards. Typical provisions regulating building height designate the range of allowed building height. In case of General Frontage Sites of this zoning code, the allowed building height is from 3 to 9 stories. By using the height parameter, a Building object can represent the range of allowed building height requirement.

The setback requirement is represented in the Building objects. The location of setback lines are defined in the regulating plan. The minimum distance from the setback line to the building envelope is defined in the Siting section of the building envelope standards. By using the setback parameter, the envelope of Building object can be position on the allowed setback line.

In PURMs, the AssignBuildingCodes application updates the parameters for the building codes. Such parameters are built in the multiple Building objects and the application updates all of them simultaneously. Figure 40 shows the initial parameter settings of PUDM. Once an operator

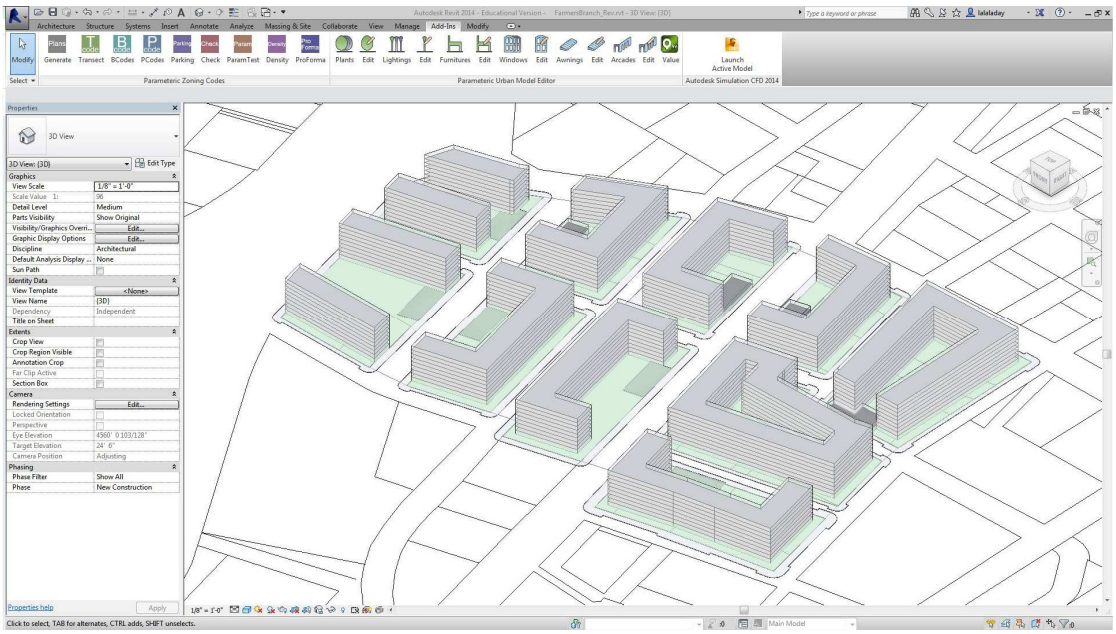


Figure 42. The PUDM is updated according to the parameter changes

selects a particular Building object, the application can perform the parameter update process. A user interface displays the parameter settings of the selected Building object and allows the operator to change the value as required. Then the new parameter values can be applied to the other Building objects at a same time (Figure 42).

Based on the selected Building object, the application locates the other Building objects within the same transect zone, reads the user input values from the user interface, and finally updates the parameter values. A detailed process is explained in Chapter 5.

6.3.4 Update the PUDM

When a provision has conditional or relational constraints relating to other provisions, a relationship among provisions is found. When such provision is modeled as a parameter in the PURMs, the parameter value can be determined by other parameters. In this section, I will demonstrate how the related parameters can be updated by the PUDAs.

Updating Parking objects is an example to demonstrate how the application establishes the parameter relationships among the PUDOs. In the following section, I will describe how the parameters of Parking objects are determined by other parameters of Parcel and Building objects.

6.3.4.1 Assign the parking codes

The parking standards in this zoning code have provisions specifying the minimum parking spaces according to the building floor area and use, which are regulated by the building envelope standards. For instance, residential buildings shall provide at least 1 and 1/8 parking space per residential unit. Parking space of non-residential building is determined by the gross floor area.

The position of the parking structure is regulated by the parking setback lines in the regulating plan and the parking setback regulations in the building envelope standard.

The siting section of the building envelope standards has provisions for the on-site open space. For instance, a contiguous open space shall be provided at least 15% of the total buildable area. In the definition section, the buildable area is defined as the lot area that building may occupy behind the RBL and setbacks. In the regulating plan, the RBLs are on the parcel boundary, so the parcel area is used as the buildable area.

As running the AssignParkingCodes application, a user interface is opened to ask regulatory information for the parking codes. Currently, the application collects information of the open space ratio, the minimum number of parking space for the floor area, the available street parking spaces, and an average area per one parking space (Figure 43). Then the application stores the values into the parameters of the Parcel, Building, and Parking objects. The parameter values will be used to update geometry of the Parking objects.

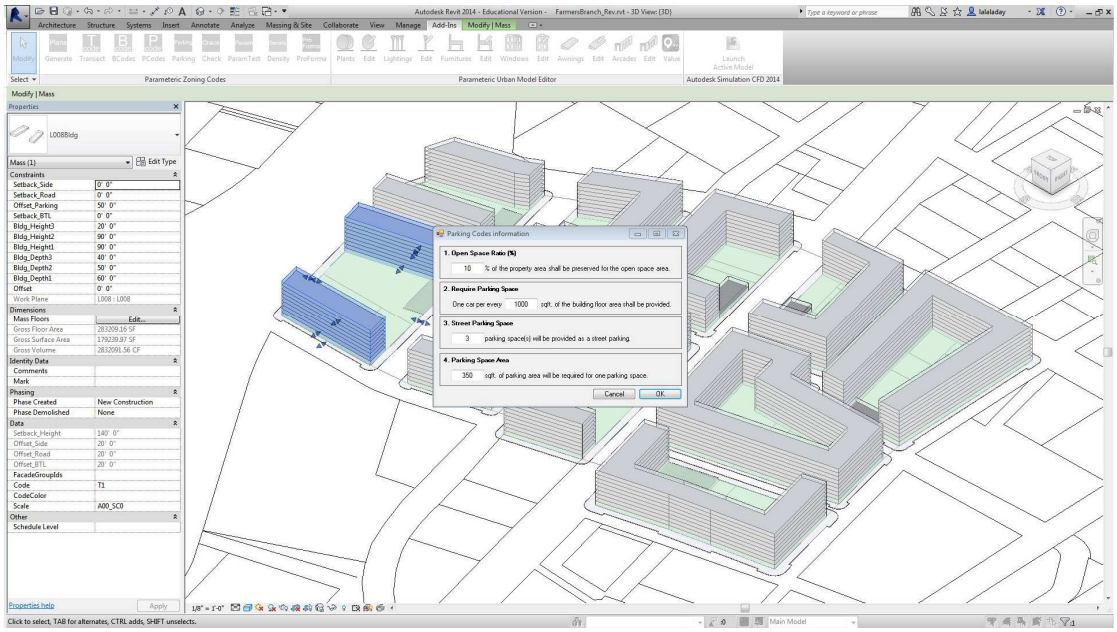


Figure 43. The AssignParkingCodes application. Note: The parameters of parking regulations are assigned by the AssignParkingCodes application

6.3.4.2 Update the Parking objects

In the urban regulations, some provisions regulate form and scale of the parking structure, but they are more flexible than the provisions controlling the front façade location. Therefore, geometry and location of the Parking objects would be less conclusive than the Building objects. For instance, a Parking object is currently attached on the Building object surface, there are numerous positions allowed by the urban regulations. For the flexible position and dimension, three parameters are built in the Parking objects.

Updating geometry of the Parking objects is performed by the predefined rule explained in Chapter 5. The first step is calculating the required parking spaces according to the regulatory information and the gross floor area of buildings. Second, the available lot area for parking is calculated by using the parcel area, the building footprint area, and the open space area. Lastly, the footprint area and the number of floors of the parking structure are determined.

The Parking object and the UpdateParking applications enable testing the constraints among parcel, building, and parking. As the building height increases, the required parking spaces increases. The maximum building height in this zoning code is 9 stories as shown in Figure 44. By running the UpdateParking application, the height of parking objects could be updated. In some lots having high density, the parking structures became taller than the main building. In that case, the application shows a message that the parking structure shall be lower than the main building, which is defined in the height section of the building envelope standards.

The gross floor area of the Parking object is constrained by the Building object. The footprint area of the Parking object is determined by the footprint area of the Building object, the open space area ratio, and the area of the Parcel object. In contrast, the building height is constrained by the parking structure height when the parking structure becomes higher than the building. In

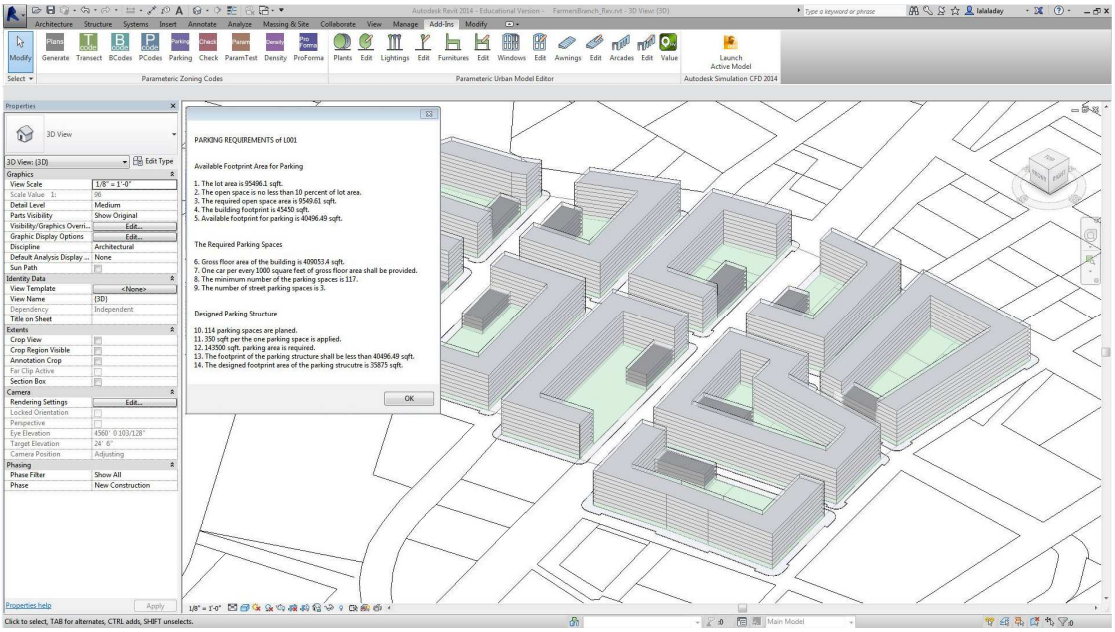


Figure 44. The UpdateParking application. Note: The UpdateParking application updates the Parking objects according to the Building objects and the parking regulations.

sum, the relationships among constraints could be visualized and tested in the PURMs.

6.3.5 Analyze the PUDM

Until now, I created the PURM for the zoning code of Farmers Branch and assigned a set of parameters. In this section, I will demonstrate how the PURM can be analyzed by the PUDAs. The analysis criteria are zoning code compliance, development density, economics, and energy use and cost. The energy consumption is analyzed by using a cloud based energy simulation with Green Building Studio. Other analyses are performed by the PUDAs.

The PURMs could have the analytical capabilities by creating an urban regulation model in Parametric BIM. In addition, the various functionalities of Revit became usable in the PURMs.

6.3.5.1 Check code compliance

When a provision regulates form and scale, the regulation constraints can be stored in the

PUDAs as a condition for code compliance checking. According to a provision of the building envelope standards, the height of parking structure shall not be higher than the main building height. In the UpdateParking application, the Building object height and the Parking object height are compared. When the Parking object is taller than the Building object, a message box displays the code compliance result.

When a set of parameters are assigned to the model, geometry of the PUDM is updated. Often, two solid objects can be overlapped or intersected each other. For instance, if a large footprint area is applied to the Parking object, the Building object and the Parking object can be intersected. In this step, the collisions among the objects are detected by using the CheckCollision application. When a collision is discovered, the application changes the color of the Parcel object to visualize the problematic sites.

Even though I explained two applications, Parametric BIM may have a potential to convey the various code compliance checking processes. In Parametric BIM, the coordinate information of face, line, and point is identifiable. The distance between them can be also measured. In addition, Revit API provides many methods such as finding intersecting objects, reporting the intersecting point information, and measuring the closest distance from one to the other object. If a provision regulates the minimum and maximum distance between two elements, an application would be created to carry out regulation compliance checking.

6.3.5.2 Analyze density

In this zoning code, the height control is the main criteria for regulating development scale. Some provisions control development density to prevent superblock design such as a large building footprint area and long linear building façade. No provisions control the maximum or minimum development density.

In the PURMs, the object attributes for the FAR calculation can be retrieved. The lot area is obtained from the Parcel object, the gross floor area of building is obtained from the Building object, and the FAR is calculated by the application.

Once the FAR value of each lot is calculated, the value is written to the Parcel object. The overall development density is stored in the Site object. Then the user interface displays information of development density. The density distribution compared to the average density is visualized with the color codes. The scroll bar in the user interface changes the number of color codes. When two colors are applied, the sites above average density turn out blue and the sites below density turn out magenta. Figure 45 shows the distribution of development density with 10 types of color codes. The visualization of development density would enable the code makers to predict the most or the least dense developments.

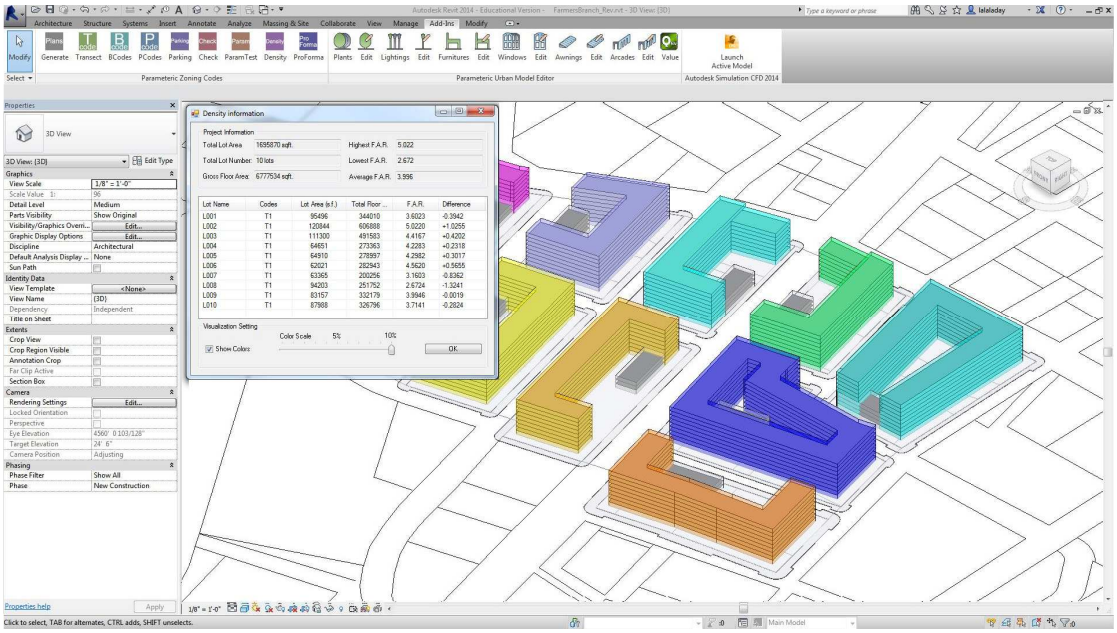


Figure 45. Analyze development density. Note: The development density of each lot is visualized with a color code.

By using the application, the most and the least dense area can be visualized. The distribution of development density varies over the parameter values, but a deviation exists even though the identical parameter values are applied.

6.3.5.3 Perform economic analysis

In this section, I experiment the economic analysis by using the ReportModelInfo application and the ReportProForma application. From the PUDM, I retrieved the geometric attributes and the building use. As presented in the previous section, development density was calculated. The ReportModelInfo application enables exporting model information of the PUDM to excel spreadsheets. In the spreadsheet, both the object parameters related with the urban regulation provisions and the object information such as density and the gross floor area are included. The values in the spreadsheet are used in the Pro Forma estimation. The ReportProForma application performs BIM-based economic analysis by using the simplified Pro Forma analysis methods described in Chapter 5. In the Pro Forma estimation report, project information, total project cost, NOI, gross revenue, and development profit are analyzed.

Many urban regulation provisions designate a range of allowances such as maximum and minimum building height. For instance, the building height in General Sites shall be between 3 and 9 stories. In a single run, ReportProForma carries out multiple economic analyses for testing 7 types of the building height parameter. In this demonstration, 23 types of model information reports and the Pro Forma reports are generated for different parameter sets.

6.3.5.4 Perform environmental analysis

The energy analysis is carried out with the energy simulation features in Revit, which is a cloud-based energy simulation to analyze energy use and cost, potential renewable energy, annual carbon emissions, monthly cooling and heating loads, etc. The energy simulation settings can be

edited in the Revit interface such as the type of buildings, constructions, and HVAC systems (Figure 46). Some values are related to the urban regulations, but others are required only for the energy simulation. Then an energy model is generated from the PUDM by using the “Enable Energy Model” function. The energy model is synchronized with the PUDM. After any changes are made in the urban model, the energy model is automatically updated. After completing the energy simulation in cloud, the simulation result report is opened in the window (Figure 46).

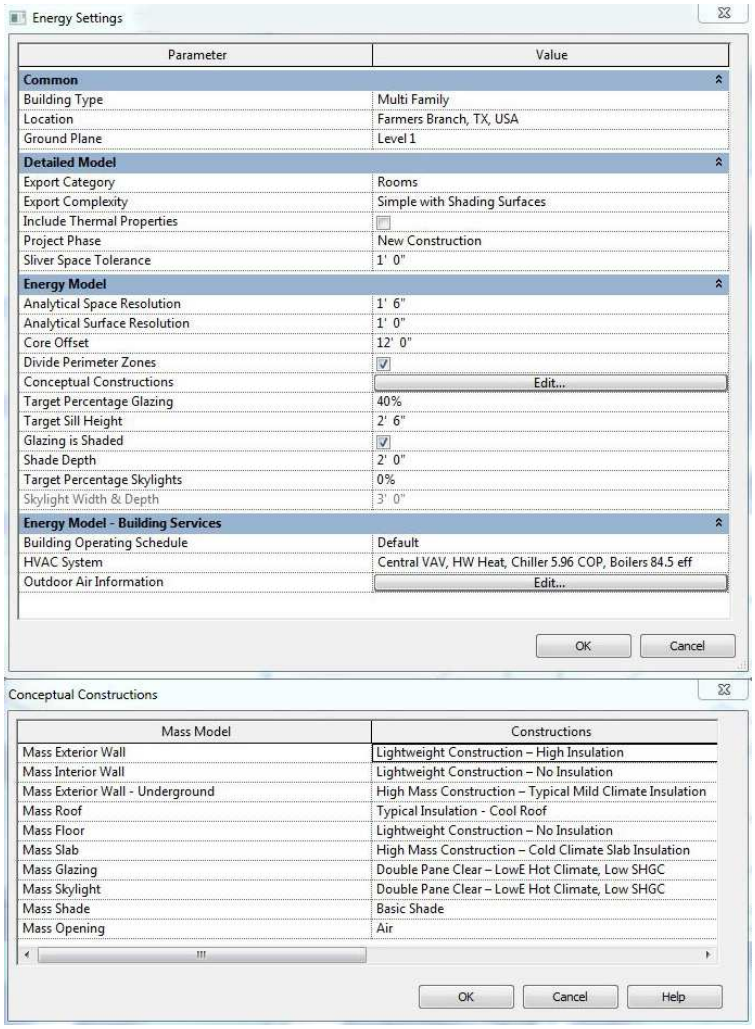


Figure 46. The energy simulation setting window.

6.4 Observations

In the experiment, a set of parameter values was applied to the PUDM to test a range of regulatory provisions. Among them, I exemplify six test cases to demonstrate how multiple zoning code schemes can be tested and analyzed in the PURMs. It is required to remind that the main scope of this demonstration is not to achieve accurate analysis results but to implement the potential analysis process in the PURMs.

The selected schemes have 3 types of building height and 2 types of building depth. In the building height provision, the minimum height is 3 stories and the maximum height is 9 stories. I randomly selected the minimum, the medium, and the maximum values as building height. In addition, I applied 50 feet and 60 feet as the building depth for the multi-family houses. In the zoning code, the building front façade needs to be located on the RBL lines, so I applied 0 feet as the building setback.

Table 19 shows information of the six test cases. First, I listed major provisions guiding the experiment. Second, main parameters applied to the PUDM are summarized. Then the results of the development density analysis, the economic analysis, and the energy simulation are compared.

In the density analysis, the development scale is analyzed such as building footprint area, building gross floor area, and F.A.R. are calculated. The F.A.R. values are between 1.2 and 4.5. In the economic analysis, the development profit is predicted by using the Pro Forma estimation method. In this table, I presented a part of the calculated values. In the energy analysis, energy performances of the development schemes are analyzed such as energy use and cost, annual carbon emissions, and the life cycle energy cost, which are a part of the simulation results.

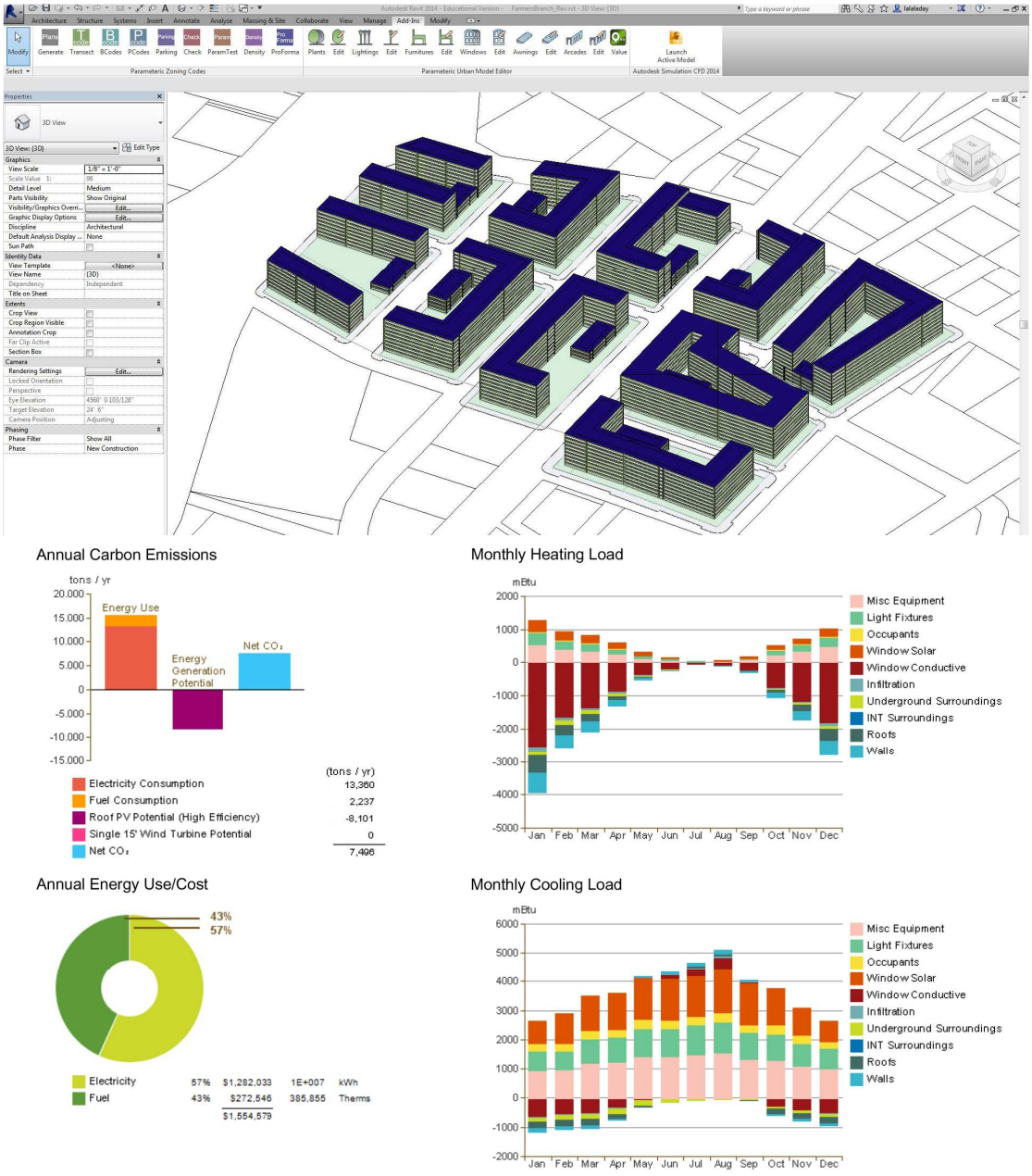


Figure 47. The energy model and the simulation results. Note: The energy model is generated on the PUDM. Then energy simulation results are displayed in the Revit interface.

Table 19. The test cases in the experiment

Category	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
Major provisions						
• Building height	The building height shall be between 3 and 9 stories.					
• Building disposition	The building front façade shall be located in the RBL.					
• Parking disposition	The parking structure shall be located behind the parking setback lines.					
• Open space	10 % of the Parcel Object area shall be provided as on-site open space					
Main parameters						
• Building envelope type	T1 (General Frontage)					
• Open space ratio	10 % the Parcel Object area					
• Height (floors)	3	3	6	6	9	9
• Depth (ft.)	50	60	50	60	50	60
• Setback (ft.)	0	0	0	0	0	0
• Building use	Mixed use (Retails in the ground floor and residential in the upper floor)					
• Parking space ratio to the gross floor area	1 parking space per every 1000 sq. ft.					
• Parking space area per car (sq. ft.)	300					
Density Analysis						
• Site Area (sq. ft.)	1,193,493					
• Total Parcel Area (sq. ft.)	847,938					
• Building foot print area (sq. ft.)	374,808	437,735	374,808	437,735	374,808	437,735
• Building gross floor area (sq. ft.)	1,060,182	1,238386	2,184,608	2,551,600	3,309,032	3,864,809
• F.A.R.	1.2	1.4	2.5	3.0	3.9	4.5
Economic Analysis						
• Gross Revenue (1000$)	40,551	47,368	83,561	97,598	126,570	147,828
• Net Operating Income (1000$)	35,251	42,068	78,261	92,298	121,270	142,528
• Capitalization Rate	0.085	0.085	0.085	0.085	0.085	0.085
• Sale Price (1000$)	414,728	494,925	920,720	1,085,867	1,426,711	1,676,811
• Project Cost (1000$)	414	482	847,041	988,333	1,279,944	1,493,918
• Development profit (1000$)	591	12,175	73,679	97,533	146,766	182,892
Energy Analysis						
• Annual energy use/cost (1000$)	1,180	1,351	2,463	2,802	3,722	4,237
• Energy Use Intensity (kBTU/sq.ft./year)	55	54	56	52	55	52
• Life Cycle Energy Cost (1000$)	16,079	18,406	33,552	38,173	50,700	57,717
• Annual carbon emissions (tons)	5,764	6,692	17,852	20,689	29,879	34,646

6.4.1 Density analysis

Figure 48 shows the comparison of the F.A.R. changes of six cases having three types of building height and two types of building setback. Generally, a similar order of the F.A.R. values among 10 lots is found across the cases. The lot 2 shows highest density between 1.7 and 5.8, while the lot 10 has lowest values between 0.5 and 1.7 for all cases. The deviation increases from the case 1 to case 6. In the case 1, the deviation is about 1.2, but it increases to 4.1 in the case 6. The incremental level is dependent on the lots, but the lot 2 increases mostly.

In sum, once a lot shows higher density than others, the highest density may be discovered at the same lot regardless of the height and the depth parameters. A steep slope in the plot implies the higher impact of building height and depth on the development density.

The ratio of ground floor area to lot area is related to the F.A.R. value, because the Building

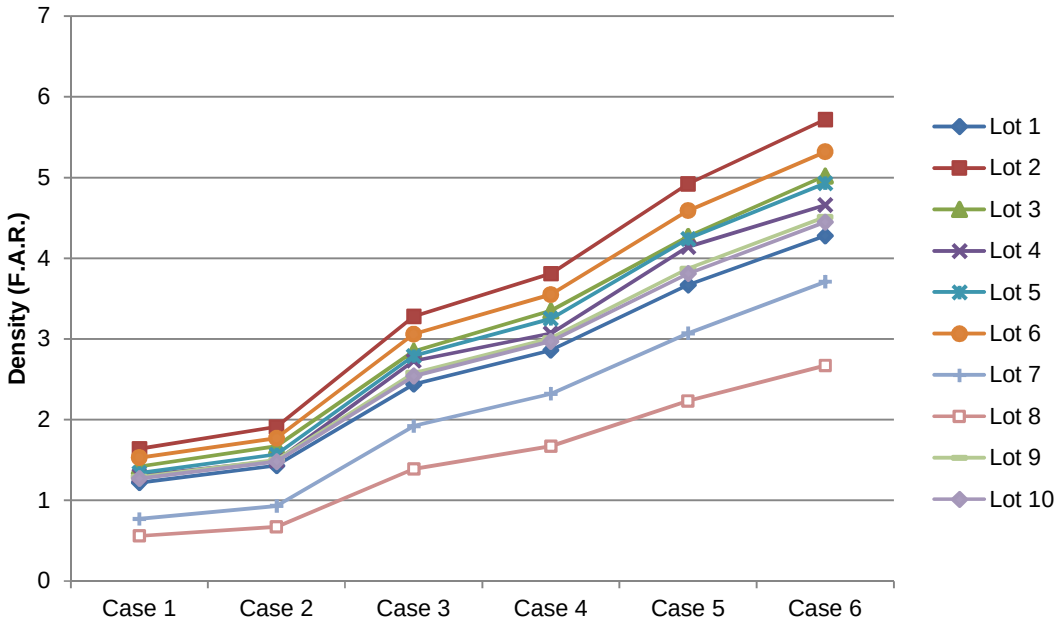


Figure 48. The density distribution of six test cases

Objects is a solid extrusion of the building footprint. The ground floor area is changed not by the height parameter but by the depth parameter. In Figure 49, the case 1, 3, and 5 as well as 2, 4, and 6 have the same building depth, so they show the same distributions respectively. The ratio value varies over the last, but a similar order is consistently discovered across the cases. In addition, the order in the ratio values in Figure 48 is identical with the order of the F.A.R. values in Figure 49. The ground floor area is related to the RBL lines that vary over the sites.

In the PURMs, various model parameters and quantitative values can be used to analyze the correlations between the urban regulation provisions and development capacities. Once the correlations are discovered, it has potential to predict the impact of the urban regulation provisions on the development capacities.

6.4.2 Economic analysis

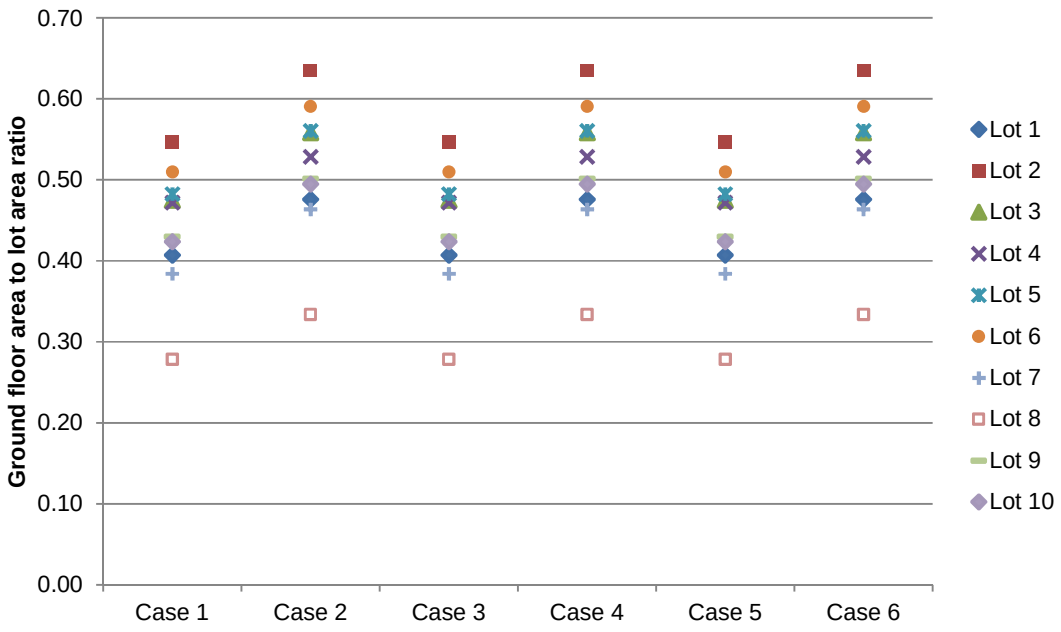


Figure 49. Density distribution of six test cases

Figure 50 shows the economic analysis results listed in Table 19. The plot presents a part of analysis results including the project cost, the sale price, NOI, gross revenue, and the development profit. The analysis is carried out by using the Pro Forma estimation method described in Chapter 5 and the ReportProForma application. To do so, supplementary values that are not defined in the zoning codes need to be stored in the parameters such as the development costs, the occupancy rate, and the capitalization rate. These values affect the analysis results.

For instance, the efficiency rate of lettable space is required to estimate rentable space area from the building floor area. As the efficiency rate is changed from 0.85 to 0.75, the development profit is decreased as shown in Figure 51 and 52.

In the PURM, the variables for economic analysis are stored as the object parameters and the Pro Forma estimation is carried out by using the parameters. Therefore, a range of economic variables can be tested to estimate the potential development profits.

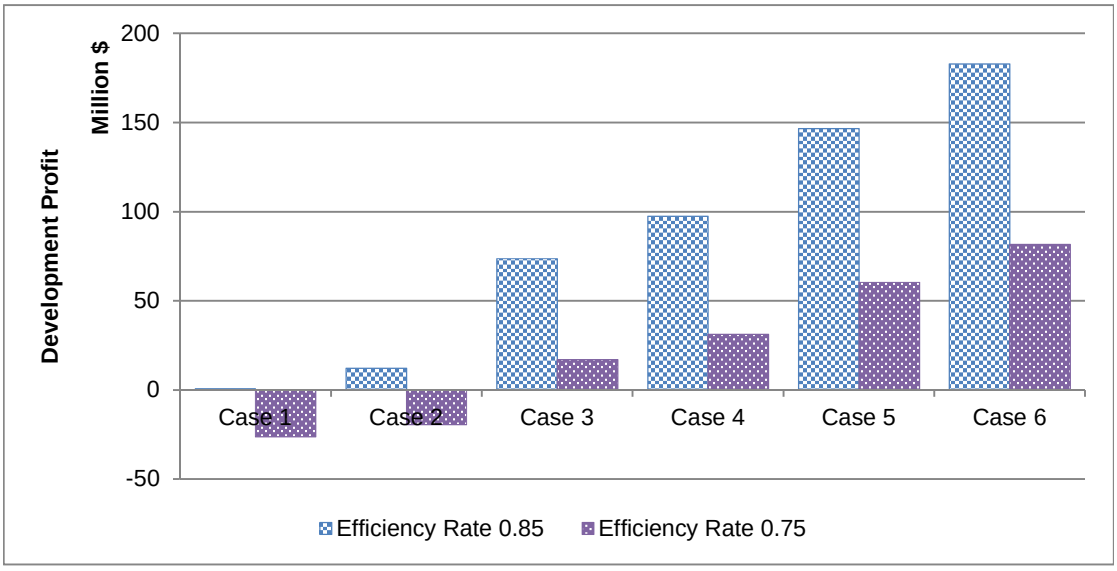


Figure 50. Development profit of six test cases

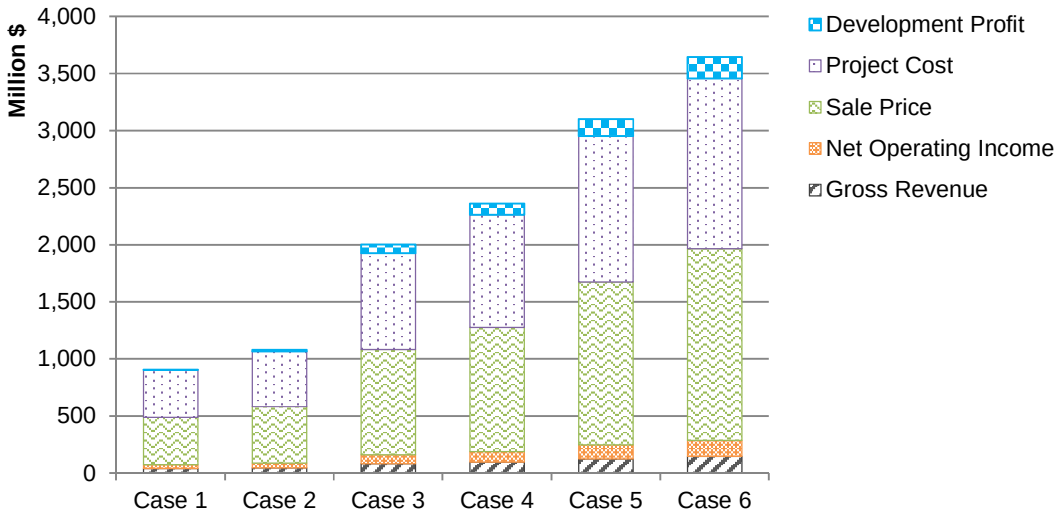


Figure 51. Economic analysis results with the efficiency rate 0.85

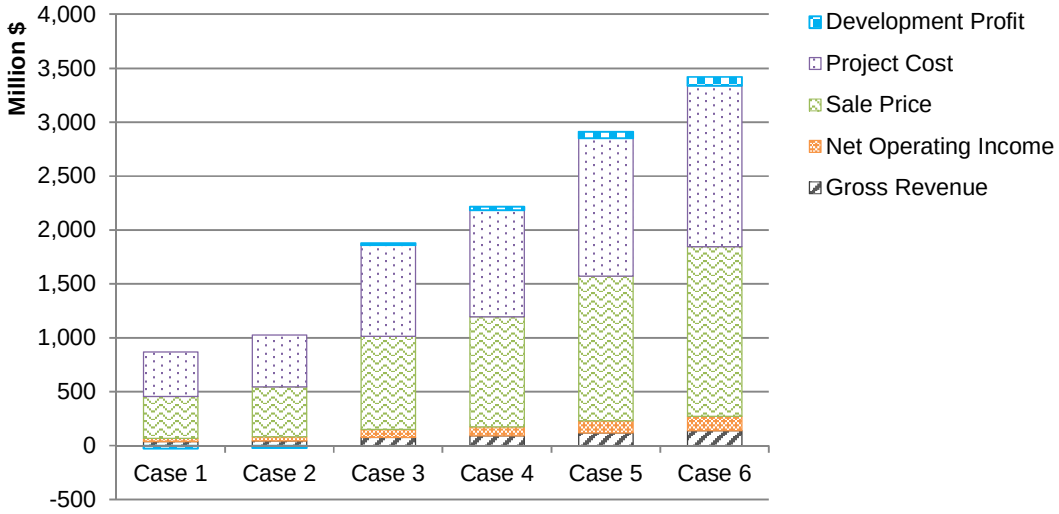


Figure 52. Economic analysis results with the efficiency rate 0.75

6.4.3 Energy simulation

In Figure 53, I presented the life cycle energy cost with development profit listed in Table 19. The life cycle energy cost is an accumulated cost of the 30 year energy costs, while development

profit is returned after completing leasing or selling. In the case 1 and 2, the life cycle cost is higher than development profit. The case 6 shows the highest values in the life cycle cost and development profit. In addition, the highest value of annual carbon emissions is found in the case 6.

Currently, there are many limitations for the zoning code analysis, but the BIM-based energy simulation is rigorously developing. For the most, it is required to improve the accuracy in the energy model creation and to provide a reliable simulation capability for the urban scale projects.

Even though I explained a simple comparison, it implies that energy simulation of the zoning code schemes may be useful to predict the environmental footprint that are critical information for the resource management.

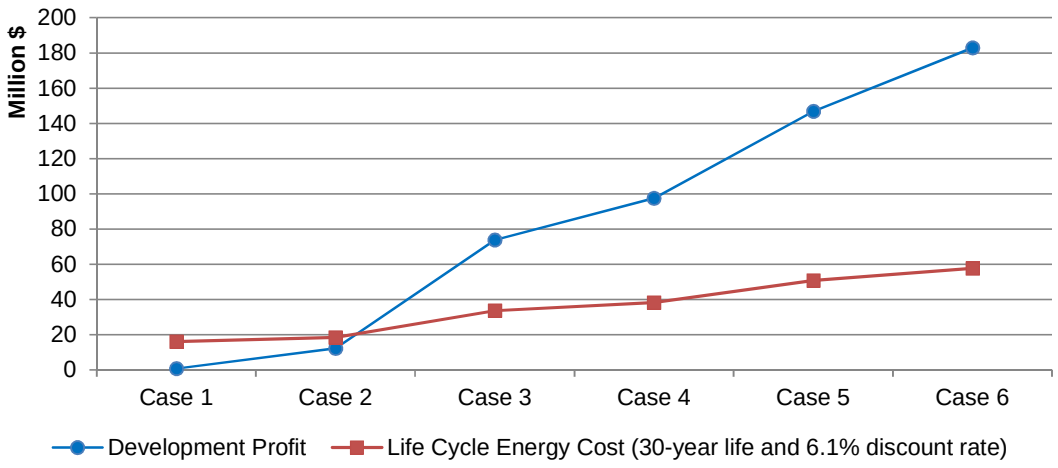


Figure 53. Development profit and life cycle energy cost

6.4.4 Operation time

The application running time is measured. In the experiment, I used a pc having Quad core 1.73

GHz processor, 8 GB system memory, 2GB Video memory, and Windows 7 64-bit operating system. The following table shows the operating time of each task by using PUDAs.

The modeling was processed in three days and I measured the modeling time for each objects: 245 minutes for the Block objects, 283 minutes for the Building objects, 37 minutes for the Parcel objects, and 11 minutes for the Parking objects. As more parameters control geometry, the longer modeling time was spent. The Block and the Building objects are more complex than the Parcel and the Parking objects. The learning process also affects the modeling time. I modeled

Table 20. An operating time of the PURM

Tasks	Tasks	Application	Time
Create PUDOs	Create Site object	Revit	5 min
	Create Block objects	Revit	245 min
	Create Parcel objects	Revit	283 min
	Create Building objects	Revit	37 min
	Create Parking objects	Revit	11 min
Create Parametric Urban Model	Assembles individual PUDOs to create a Parametric Urban Model	GeneratePlan	8 min
	Create arrays of trees or lighting fixtures on the Block objects	GenerateArray	34 s
	Edits the position, spacing, and the object type of the arrays	UpdateArray	10 s
Assign parameters	Update parameters in Parcel objects (transect codes)	AssignTransectCodes	10 s
	Update parameters of Building objects	AssignBuildingCodes	15 s
Update the Parametric Urban Model	Update parameters of Parking objects (parking requirements)	AssignParkingCodes	10 s
	Update geometry of Parking objects	UpdateParking	25 s
Analyze the Parametric Urban Model	Detect collisions among PUDOs	CheckCollision	5 s
	Analyze and visualize density of the parcel and the project level.	VisualizeDensity	5 s
	Analyze project Pro Forma and generate reports.	ReportProForma	2 min 30 s[1]
	Analyze environmental performances	Revit	7 min 20 s[2]

1. The economic analysis time is measured for nine design schemes.
2. The energy simulation time is measured for one design scheme. The total time for nine design schemes were not measured, but it took over 5 hours including the energy model generation in Revit and energy simulation in the green building studio cloud.

the objects in a sequence of Block, Building, Parcel, and Parking. Among ten Block objects, modeling for the first object took about two hours, while nine objects took about two hours. Among ten Building objects, the first object took about two hours.

Overall, the PUDAs enabled fast urban regulation modeling. For instance, the GeneratePlan application can assemble a series of the objects and create the PUDM, which is significantly faster than the manual assembling process. In addition, the GenerateArray and the UpdateArray applications support the modeling process of the streetscape with the street trees and lighting fixtures. In this experiment, I could assemble the individual objects for 10 lots in short time by using the applications.

In the next chapter, I will elaborate three research contributions and encapsulate this research with research implications, originality, reliability, validity, and future works.

7 CONCLUSIONS

This research has produced three contributions to knowledge: represent urban regulation provisions in parametric BIM with the PUDM, test development performances of the candidate zoning provisions with the PUDAs, and reduce ambiguity in interpretation of the urban regulation provisions with the PURMs.

In the following sections, I will summarize the overview of this research. Next, I will elaborate three contributions with the motivation and proposition, the methods and evidences, significances, and limitations. Lastly, I will encapsulate three contributions with research implications, originality, reliability, validity, and future works.

7.1 Summary

In this dissertation, I have developed the PURM for predicting development performances of the urban regulation provisions. In particular, I created the PURM that consists of the PUDM and the PUDAs. I carried out the research through following five research phases.

Review of literature on urban regulations: I conducted a literature review on the context of urban regulations and zoning in the United States, computational urban regulation modeling, and the emerging parametric BIM technologies for regulation modeling.

Analysis of recent zoning code samples: I reviewed four contemporary zoning codes to identify the common types of structures, components, and provisions that were investigated in the urban regulation modeling in BIM. Based upon the results, I formulated a set of zoning code components and their parameters as well as an object hierarchy among the code components.

Parameterization of the urban regulations with PUDM: I developed the PUDOs and the PUDM

in BIM. I represented form implications of the textual urban regulation provisions with the PUDOs and assembled them to create the PUDM.

Development of PUDAs: I developed software prototypes of the PUDAs. The PUDAs established relationships among the PUDOs and their parameters. In addition, the PUDAs performed the complex operations for the urban regulation modeling in parametric BIM.

Experiment and evaluation: I conducted an experiment with an existing zoning code by using one of four zoning codes analyzed in the second phase. I established an experimental scenario to demonstrate the urban regulation modeling process with the PURMs. The experiments provided empirical evidences for the research propositions.

7.2 Represent Urban Regulation Provisions in Parametric BIM with the PUDM

This is the first contribution of this research, which was investigated in Chapter 4, Parameterize urban regulations, and Chapter 6, Experiments and Evaluations. In PURMs, form implications of urban regulation provisions are represented as 3D objects in BIM with objects' parameters.

7.2.1 Motivation and proposition

Many provisions designate a range of allowances, so an urban regulation model in BIM needs to express a range of form implication. The parametric modeling approach in BIM is a motive that initiated this research. In parametric BIM, a geometrical attribute of a BIM object can be assigned through its parameters so that geometry of the BIM object can be manipulated by parameter change. The textual provisions of the urban regulations significantly differ in semantics of the objects in BIM. However, there has been no comprehensive research on the urban regulation modeling in parametric BIM for the United States urban regulations. Therefore, I formulated the following proposition.

- A PURM based on parametric BIM can represent common, typical, and important provisions of the urban regulations.

This proposition was addressed in the step 1 and 2 of the experimental scenario. The observation results in the experiments provided strong empirical evidences.

7.2.2 Methods and evidences

The form and scale of the PUDM could represent a part of form implications of the urban regulation provisions. The Building Objects of the PUDM represent the provisions of the RBL, setbacks, building height, and building use in the regulating plans and the building envelope standards.

The streetscape standards specified the configuration of streetscape design such as the traffic lanes, on-street parking, sidewalks, and street trees. The Block Objects of the PUDM represent a part of provisions of the streetscape standards such as the dimension of the sidewalks, on-street parking, and the street trees (Figure 40).

The Building Objects of the PUDM represent the range of regulatory constraints such as the minimum and the maximum building height. The parametric modeling capability in BIM enables a Building Object can represent a set of regulation constraints. For parameter manipulation of the Building Objects, the AssignBuildingCodes application enabled to update the parameters of ten Building Objects simultaneously. The value of building height, depth, and setbacks were tested in this experiment.

Some provisions that do not directly affect urban form are built in the PUDOs when the provisions are related to other provisions that directly change urban form and scale. For instance, the Parcel Objects have a parameter of the minimum open space ratio. The parameter value is

required to determine the available footprint area of the Building and the Parking Objects. The transect codes are required to determine the type of building envelope standards that affect form and scale of the Building Objects. The Block and the Parcel Objects were created to store the transect codes as their parameters. The type of transect codes were visualized with the color codes of the PUDM (Figure 41).

The parameters of the Parking Objects represent the provisions required for the parking space calculation. Often, the zoning code has the provisions regulating the number of the parking spaces based on the building use and the gross floor area. The Parking Objects has a parameter to represent such provision (Figure 43), which was used to calculate the number of parking spaces by using the UpdateParking application of PUDAs (Figure 44).

7.2.3 Significances

The PUDM shows that form implication of the urban regulation provisions can be visualized with the objects in parametric BIM. Geometry of the PUDM is critical for visualization in urban regulation modeling. Owing to the parametric modeling mechanism, the PUDM can represent a range of candidate urban regulation provisions. The custom parameters can be used to store a range of the provision values. The geometrical attributes of the PUDOs can be transformed by changing parameter values. The way of geometry change can visualize how an urban regulation provision changes form and scale of our built environment. In addition, geometry information of the PUDM provides various quantitative values required for both urban regulation interpretation and performance analyses.

The urban regulation modeling rules investigated in Chapter 4 are expected to be applied to the future urban regulation modeling in parametric BIM. To formulate the rules, I employed the concept of form, function of the Sullivan’s notion and the Clayton’s definition, and modified it

to the urban regulation modeling in parametric BIM as follows:

- Form is the geometry of the built environment resulted from the urban regulations.
- Function is the desired qualities of the built environment that the urban regulations establish.
- Behavior is the response of the built environment to the particular urban regulations.

Understanding textual urban regulations with this concept enabled me to establish the significant urban regulation modeling rules. In the PUDOs, form is objects' geometry and behavior is the way of geometry change through the object parameter. An important rule is that behavior of an urban regulation provision can be defined in diverse ways, but the object behavior can determine the level of complexity in parametric modeling. In particular, manipulating solid object geometry by using hosted void objects could ease such complexity in parametric modeling. These rules were formulated from some urban regulation provisions, so further research would be required to formalize the comprehensive rules of urban regulation modeling in parametric BIM.

7.2.4 Limitations

The parametric modeling for the PUDOs is complex. Even though I established the urban regulation modeling rules, individual objects need to be created manually. Applying multiple parameters to the BIM objects took long time. Generally, the modeling time proportionally increased according to the complexity of site topology. In this research, the site is simplified as a flat surface so I did not account for the slope of the site. To consider the slope level of the sites, more complicate modeling rules needs to be investigated.

I simplified the type of urban regulation components into five objects of site, block, parcel, building, and parking. No objects for the traffic lanes and public spaces and community parks

are included. They will be added to the PUDM.

In the PUDM, the provisions controlling urban form and scale are mainly investigated. I implemented a part of the quantitative provisions, while the most qualitative provisions are not investigated. Urban form and scale are a part of urban regulation scopes, while the qualitative provisions also have a significant role in creating built environment. Further research would be required to account for the qualitative provisions for the more comprehensive urban regulation modeling.

7.3 Test Development Performances of Urban Regulations with the PUDAs

This is the second contribution of this research, which was investigated in Chapter 5, PUDAs, and Chapter 6 Experiments and Evaluations. Four types of development performances were tested in this research including urban form, development scale, economic performances, and energy performances.

7.3.1 Motivation and proposition

One of the advantages of urban regulation modeling in parametric BIM is that the geometrical attributes of the BIM objects can be manipulated by parameter change. Once I succeeded in representing form implication of urban regulations with the PUDM, it showed a huge potential to use a wide range of BIM technologies in the urban regulation development process.

Various analytical tools are usable for the BIM-based performance simulation. The OOP-based software development environment in the BIM authoring tools can be used to perform the customized analysis processes. I envisioned that use of these technologies may enable to assess various development performances of urban regulation provisions in parametric BIM, so I addressed the following proposition.

- A PURM based on parametric BIM can enable tests of performances of the candidate urban regulation provisions before urban regulation adoption.

This proposition was addressed in the step 4 of the experimental scenario. The observation results and the simulation results provided strong empirical evidences.

7.3.2 Methods and evidences

In the step 4 of the experimental scenario, I analyzed development performances in four aspects: urban form, development scale, economic performances, and energy consumptions as follows:

- Urban form: The geometry of the PUDM represents a part of the form implication of the urban regulation provisions. As I described, a set of parameter combinations were fed into the PUDM by using the PUDAs.
- Development scale: The analysis of the density distribution was performed to demonstrate how the potential development scale such as the gross floor area and F.A.R. are predicted by the PURMs.
- Economic performances: As an example of economic analysis in the PURMs, I carried out the Pro Forma estimation by using the simplified Pro Forma estimation method and the ReportProForma application in the PUDAs. I added supplementary information for the Pro Forma estimation into the parameters. Then I calculated the gross revenue, NOI, the project cost, the sale price, and the development profit.
- Energy consumptions: I carried out energy simulation by using the energy simulation analysis capabilities in Revit that utilizes the green building studio, a cloud based energy simulation system. The simulation results include the annual and the monthly energy use and cost, Energy Use Intensity (EUI), the life cycle energy cost, and the annual carbon emissions.

During the analysis process, I utilized a set of the PUDAs. The VisualizeDensity application is used to calculate the development density and to visualize the density distribution with the color codes in the PUDM. The ReportProForma application is implemented for the BIM-based economic analysis.

Urban form is visualized with the PUDM. Owing to the parametric modeling mechanism in BIM, the PUDM could transform its geometry according to the parameter update. The Building Objects represented a range of building height, setback, and depth. The Block Objects represented a range of sidewalk widths, on-street parking spaces, and the street trees. The Parking Objects represented a range of the on-site parking spaces or the parking structures. For instance, in the Farmers Branch zoning codes, the minimum building height is 3 stories and the maximum is 9 stories. In the experiment, the height parameter of the Building Objects was changed to represent the 7 types of building heights. In short, the use of parameters enabled the PUDM to represent multiple candidate values of the urban regulation provisions.

The analysis of the density distribution across the ten sites (Figure 45) demonstrates an example of how the development performances can be predicted in the PURM. I could monitor the density distribution of ten lots and estimate the average development density. In addition, the impact of the parameter values on the development density of each site could be understood.

In the PURMs, a loop of model changes can be automatically repeated as required. For instance, the economic analysis in this demonstration tested 6 schemes with 3 types of building height and 2 types of building depth at a same time. In a single run of the ReportProForma application, the Pro Forma reports and the model information reports were generated for six schemes simultaneously.

7.3.3 Significances

The significance of this contribution is that the urban regulation modeling in parametric BIM can enable the assessment of the urban regulation effect on the built environment in the urban regulation development process. I successfully implemented the PUDAs to analyze a part of development performances including development capacity, the feasibility of development, and energy performances.

Further studies may be required to improve the result accuracy to be used in practice, but the analysis process was so fast that a set of urban regulation schemes could be tested in short time (Table 20). It implies that an immediate assessment during the urban regulation development can be conducted to ensure the sustainable built environments.

7.3.4 Limitations

The PUDAs implemented limited tasks that I formulated in Chapter 4. The developed applications support a part of urban regulation modeling in parametric BIM. Still there are many difficulties and problems in urban regulation development that can be supported by the PUDAs.

Both the economic and the energy analyses were performed not in the parcel level but in the project level. The simplified Pro Forma estimation carried out the project level analysis. In the energy simulation in Revit, the all building objects are considered as one, so the parcel level simulation results cannot be obtained. In cases of economic analysis, the parcel level analysis results can be obtained by modifying the Pro Forma estimation method.

In environmental analysis, however, further limitations exist. The limited type of building use and materials are allowed in the energy simulation, so there are mismatches between the urban regulation model and the energy model, which makes the simulation results less accurate.

The application development of PUDAs was carried out by using particular software, so many problems and solutions found in the application development, which are software dependent. Revit is a tool for the AECO industry, so most functionalities are designed to be used in architectural design. The modeling and application development in this research focused on the urban scale project, it is required to overcome various limitations of the software environment. As a result, some findings are not generalizable to other BIM software environment.

The maintainability of the PUDAs is related to the software dependency of the application development. Developing the applications with the particular BIM authoring tools enabled fast implementation and visualization, but a continuing revision would be required as the tools and their API are updated.

7.4 Reduce Ambiguity in Interpretation of Urban Regulations with the PURMs

This is the third contribution of this research, which was investigated across the experimental scenario. In particular, I experimented how the PURMs can reduce ambiguity in interpretation when multiple provisions are related to each other.

7.4.1 Motivation and proposition

Ambiguity in zoning code interpretation is one of the critical challenges in design. Urban form and scale are frequently regulated by multiple provisions. Many provisions are related to each other and the relationship among the provisions makes interpretation complex. If the object parameters in BIM can represent such relationships among the urban regulation provisions, some parameters can be automatically determined by others. In addition, when a particular site condition is considered in urban regulation interpretation, it becomes more complicated. To address these issues, I addressed the following proposition.

- A PURM based on parametric BIM can reduce ambiguity in urban regulation interpretation during the design phase.

This proposition was addressed across the experimental scenario.

7.4.2 Methods and evidences

When a provision is related to other provisions, interpretation of the urban regulations can be complicated. As I described, some relationships among the provisions can be captured as the relationships among the parameters of the PURMs. Once the parametric relationships are established, the parameter values can be synchronized. It implies that once one parameter value is decided, the other associated values can be determined.

For instance, the urban regulation has a provision regulating the minimum open space area ratio to the site area. In the PURMs, the Parcel Object stores the ratio value into the parameter and calculates the minimum area of the open space.

Another provision in the urban regulations regulates that the height of the parking structure shall not be higher than the building height. In this condition, the height of the parking structure is associated with the building height. The UpdateParking application compares the Building Object with the Parking Object and then determines whether the Parking Object is taller than the Building Object. In this case, the relation among two heights was stored as a code compliance checking rule in the application.

In short, once some relationships among the provisions can be stored as the parametric relationships, the associated parametric values can be determined at a same time. According to the priority among the provisions, the dependency among the parameters can be defined, determining the key parameter that can affect other parameters.

Further ambiguity in interpretation is found when a set of provisions requires a sequence of interpretation. The AssignParkingCodes and the UpdateParking applications demonstrate how a sequence of urban regulation interpretation can be captured as a rule-based model transition in the PURMs. The applications estimate the required parking number, available lot area for parking, the footprint area, and the height of the parking structure. I implemented a common sequence of the parking structure design as follows:

(1) The number of parking spaces is calculated from the building use and the gross floor area.
(2) The available footprint area of the parking structure is calculated according to the site area, the building footprint area, and the open space area.
(3) The gross floor area of the parking structure is calculated according to the number of parking spaces.
(4) From the required gross floor area (3) and the available footprint area (2), the height of the parking structure is calculated.

The regulation variables of the first and the second step are often defined in the urban regulations. In the third step, I used two variables for both the average area of the parking structure per one car space and the average height of the parking structure, which are not specified in the urban regulation provisions. I assume that both variables can be identified from the architectural or engineering standards for the parking structure design (Figure 43).

As a result, once the variables were identified in the application interface, a series of parameter values were calculated and some dimensions of the Parking Objects could be calculated. Once the attributes of the Building or the Parcel Objects are changed, the Parking Objects could be updated by using the rule-based model transition. In short, (1) the parametric relationships

represent some relationships among the urban regulation provisions and (2) some interpretation sequences are represented as the geometry transition rules in the PUDAs.

In sum, it can be conjectured that the PURM enables a rule-based model transition. The transition rules can represent some interpretation sequence among the urban regulation provisions so that ambiguity in the urban regulation interpretation can be decreased.

7.4.3 Significances

The experiments of the PURMs show that ambiguity in urban regulation interpretation could be reduced in several ways.

The multiple parameters that are related to each other can be assigned at a same time by the PUDAs. In terms of urban regulations, once a value of the provision is assigned, other related values can be determined. As multiple values can be simultaneously calculated, it may be possible to reduce the amount of interpretation. In addition, it may eliminate the interpretation process to locate the all related provisions from the urban regulation documents in that the relationships among provisions can be established by and within the PUDAs.

Form implication of urban regulations is represented for all sites in the PUDM. It may shorten the interpretation process that has been carried out by individual designers. It may accelerate the code compliance checking process that has been performed by the officers.

One the other hand, the PURMs make it possible to carry out a rule-based calculation by using multiple parameters that are related to each other. When a certain calculation rule can be defined, an application can be written to locate all required parameters from the entire urban model to perform calculations based upon the rule. Furthermore, the rule among the parameters can be defined in the PUDAs, so the rule can be updated without modifying the PURMs.

7.4.4 Limitations

Typically, a prescriptive provision is less flexible. Some flexible provisions are difficult to be represented in the PUDM. In the experiment, the available parking location is more flexible than the building front façade location. Form implication of the Building Object is conclusive in comparison with form of the Parking Object.

In general, a flexible provision allows various design options, so the level of flexibility is significant information in design. Each PUDO represents multiple provisions having varying flexibility. In case of the Parking Object, the height parameter is less flexible than the disposition parameter. However, both parameters are built in the same object without considering the flexibility of the regulation provision, while it was well identified in the textual urban regulations.

This contribution was led by the single experiment with a self-test. However, the PURM has been tested in several research projects and the results have been published. The initial research findings were published in three conferences (Kim et al., 2010; Kim et al., 2011; Kim et al., 2013). In the QNV-IV system development research (Clayton, Booth and Kim, 2013; Clayton, Booth, Kim, and Zarrinmehr, 2013), the PUDM and a part of PUDAs were tested for the Campus Pointe project, mixed use neighborhood development in the Texas A&M University. The PUDMs in Chapter 4 and 5 were created based on the Campus Pointe project. Lastly, measuring reduced ambiguity with PURM would be assessed through the usability test to collect subjective responses of the user groups.

7.5 Discussion of Contributions

In the previous sections, I discussed three contributions of this research. In the PUDM, typical and important provisions of the urban regulations can be represented. By using the PUDAs, the

development performances of a range of the urban regulation provisions can be tested in the PUDM. Visualized and analyzed development performances in the PURMs can reduce ambiguity in interpretation of the urban regulation provisions. From three contributions and findings, it can be concluded that the PURM has a potential to enable the stakeholders in urban regulation development to device and use the urban regulations effectively.

The contributions of this research provide implications to further research and developments that have significant value to industry and education.

Accelerate the urban regulation modeling process in parametric BIM: The PUDAs could accelerate the urban regulation modeling process in conjunction with the PUDM. For urban regulation modeling in parametric BIM, many tasks need to be repeated across the parcels. The PUDAs could carry out such tasks without a tremendous time. A test with a set of parameters to the multiple sites could be processed fast compare to the manual process.

Predict the environmental footprint of the urban plan: Research on the BIM-based energy simulation tool is a rigorously developing area. One of increasing concerns is a seamless integration between design and energy simulation, facilitating the use of energy simulation in the design phase. In many BIM-based energy simulation tools, the energy model can be generated from BIM. It implies that once an urban regulation model is created in BIM, it may be possible to use such simulation tools in the urban regulation development process.

A shared database of urban regulation regulations: The PURMs can be shared by the urban design stakeholders for various purposes. For planning officers, they can provide the regulatory information with the PURMs. Citizens can monitor the future development in their community. Designers can access the PURMs to identify the development requirements. Different level of

access permissions may protect the significant regulatory information embedded in the shared PURM database. Some PUDAs can be provided to developers for code compliance checking prior to the planning review process.

Formalization and automation of the urban regulation modeling rules: Formalization and automation of the urban regulation modeling rules in parametric BIM is the future research. One significant finding in this research is the rules of urban regulation modeling in parametric BIM investigated in the PUDM development. The rules explained how to represent the urban regulation provisions with the BIM objects and their parameters based upon the concept of form, function, and behavior. If a set of rules can be formalized, the manual modeling process can be documented to be used in future urban regulation modeling. If the formalized rules can be mapped into the PUDAs, automation of the modeling process would be achieved.

A tool for urban regulation development: The PURMs have a potential to be used in the urban regulation development process. The PUDAs can perform fast model creation, continuing model changes via parameters, and various performance analyses of the urban regulation schemes. If a set of candidate master plans are tested in the PURMs, their development performances can be compared for decision making in planning review. In design charrette, as citizen's inputs are collected, various schemes can be immediately visualized and analyzed.

A tool for model-based research: The PURMs were developed as a tool for model-based research to provide the empirical evidences in this research. The PURMs generated the spreadsheets including a large amount of information of the urban regulation provisions and development performances. As I demonstrated in the experiment, such database can be used to analyze the relationship between the parameters and the development performances. Research on the regulation impact on the built environment is another future research by using the PURMs

and the existing urban regulations.

A tool for education: The PURMs has a potential to be used in education. In the urban design studio, the PURMs can provide zoning code information so that the students can propose their design over the PURMs. In the urban planning classes, students can understand form implication and development performances in the parametric BIM environment without using conventional land use and zoning maps.

The value of this research is that the PURMs can become a new mechanism and platform that encapsulates the urban plan instead of being a static regulation tool and that improves the quality of urban planning outcomes and development conceptualization.

REFERENCES

Adams, T. (1935). Outline of Town and City Planning: A Review of Past Efforts and Modern Aims. London: J. & A. Churchill.

Al-Douri, F.S. (2006). Impact of Utilizing 3D Digital Urban Models on the Design Content of Urban Design Plans in US Cities (Ph.D. Dissertation). Texas A&M University, College Station, Texas

Autodesk, Inc. (2014). AutoCAD Civil 3D [software]. Available from http://www.autodesk.com/products/autodesk-autocad-civil-3d/overview

Autodesk, Inc. (2014). Autodesk Revit [software]. Available from http://www.autodesk.com/products/autodesk-revit-family/overview

Autodesk, Inc. (2014). InfraWorks [software]. Available from http://www.autodesk.com/products/infraworks-family/overview

Autodesk, Inc. (2011). Project Galileo [software]. Available from http://labs.autodesk.com/utilities/galileo

Baker, N. F. (1927). *Legal Aspects of Zoning.* Chicago: University of Chicago Press.

Barnett, J. (2011). City Design: Modernist, Traditional, Green, and Systems Perspectives. Abingdon, Oxon: Routledge.

Bassett, E. M. (1922). Zoning versus Private Restrictions. *Baltimore Municipal Journal*, January 6, 2.

Baumeister, R. (1876). Town Expansions Considered with Respect to Technology, *Building Code and Economy*. Berlin: Ernst & Korn.

Ben-Joseph, E. (2005). *The Code of the City: Standards and the Hidden Language of Place Making*. Urban and industrial environments. Cambridge, Mass: MIT Press.

Ben-Joseph, E. (2009). Commentary: Designing Codes: Trends in Cities, Planning and Development. *Urban Studies*, 46(12), 2691–2702.

Bentley Systems, Inc. (2014). Bentley Map [software]. Available from http://www.bentley.com/en-US/Products/Bentley+Map/

Bettman, A. (1925). The Fact Bases of Zoning. *City Planning*, 1(2), 86–95.

Brail, R. (2008). *Planning Support Systems for Cities and Regions*. Cambridge, Mass: Lincoln Institute of Land Policy.

Carmona, M. (2009). Design Coding and the Creative, Market and Regulatory Tyrannies of Practice. *Urban Studies*, 46(12), 2643–2667.

City of Leander. (2011). SmartCode. Retrieved from http://www.leandertx.org/

City of Peoria Illinois. (2011). Heart of Peoria Form Districts. Retrieved from http://www.ci.peoria.il.us/development-codes

City of Ventura. (2011). Downtown Specific Plan. Retrieved from http://www.cityofventura.net/downtown

Clayton, M. J. (1998). A virtual product model for conceptual building design evaluation (Ph.D.

dissertation). Stanford University, Stanford, California.

Clayton, M. J. (2006). Mission Unaccomplished: Form and Behavior But No Function. Conference: Intelligent Computing in Engineering and Architecture, *13th EG-ICE Workshop 2006*, Ascona, Switzerland.

Clayton, M. J., Booth, G., and Kim, J. B. (2013). A framework for designing sustainable real estate developments using Quadruple Net Value Analysis and Building Information Modelling. The Proceedings of CIB World Building Congress 2013 conference, Queensland, Australia.

Clayton, M. J., Booth, G. J., Kim, J. B., and Zarrinmehr, S. (2013). Quadruple Net Value Analysis using Building Information Modeling and Immersive Visualization. The Proceedings of ENHSA 2013 Conference, Naples, Italy.

Clayton, M. J., Teicholz, P., Kunz, J. C., and Fisher, M. A. (1999). Virtual Components Consisting of Form, Function and Behavior. *Automation in Construction*, 8, 351–367.

Eastman, C. M., Teicholz, P., Sacks, R., and Liston, K. (2008). BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors. Hoboken, N.J: Wiley.

Eastman, C. M. (2009). Automated Assessment of Early Concept Designs. Architectural Design, 79(2), 52–57.

ESRI. (2013). ArcGIS [software]. Available from http://www.esri.com

ESRI. (2014). CityEngine [software]. Available from http://www.esri.com/software/cityengine/

Fischer, J. and Guy, S. (2009). Re-interpreting Regulations: Architects as Intermediaries for Low-carbon Buildings. *Urban Studies*, 46(12), 2577–2594.

Forsyth, A. (2003). Twentieth-Century Planning History. *Journal of Planning History*, 2(2), 181–184.

Fort Worth South, Inc. (2013). Near Southside Development Standards and Guidelines. Retrieved from http://www.fortworthsouth.org/FWS/development-standards-and-guidelines.html

Gero, J. S. (1995). The role of function-behavior-structure models in design. *Computing in Civil Engineering*, vol. 1, American Society of Civil Engineers, New York, 294–391.

Gero, J. S. and Kannengiesser, U. (2004). The situated function-behaviour-structure framework. *Design Studies*, 25(4), 373–391.

Gil, J., Beirao, J., Montenegro, N., and Duarte, J. (2010). Assessing Computational Tools for Urban Design: Towards a "city information model." The Proceedings of the 28th eCAADe conference, Zurich, Switzerland.

Göçmen, Z. A. and Ventura, S. J. (2010). Barriers to GIS Use in Planning. *Journal of the American Planning Association*, 76(2), 172–183.

Hall, P. G. (2002). Cities of Tomorrow: An Intellectual History of Urban Planning and Design in the Twentieth Century (3rd ed.). Oxford, UK: Blackwell Pub.

Hardy, D. (2005). Garden Cities: Practical Concept, Elusive Reality. *Journal of Planning History*, 4(4), 383–391.

Hascic, I. (2006). Essays on land use regulation (Ph.D. dissertation). Oregon State University, Corvallis, Oregon.

Holistic City Limited. (2014). CityCAD [software]. Available from http://www.holisticcity.co.uk/index.php/citycad

Holm, I. R. (2006). Ideas and Beliefs in Architecture and Industrial design: How attitudes, orientations, and underlying assumptions shape the built environment (Ph.D. dissertation). Oslo School of Architecture and Design, Oslo, Norway.

Hoover, H. (1926). A Zoning Primer by the Advisory Committee on Zoning: Department of Commerce. Washington, DC: Government Printing Office.

Ihlanfeldt, K. R. (2004). Introduction: Exclusionary Land-Use Regulations. *Urban Studies*, 41(2), 255–259.

Imrie, R. and Street, E. (2009). Regulating Design: The Practices of Architecture, Governance and Control. *Urban Studies*, 46(12), 2507–2518.

Jacobs, J. (1992). *The Death and Life of Great American Cities*. New York: Vintage Books.

Jorge, G., Jose, B., Nuno, M., and Jose, D. (2010). Assessing Computational Tools for Urban Design: Towards a "city information model". The Proceedings of the 28th eCAADe conference, Zurich, Switzerland.

Kim, J. B. and Clayton, M. J. (2010). Support Form-based Codes with Building Information Modeling – The PUDM Case Study. ACADIA 10: LIFE in:formation, On Responsive Information and Variations in Architecture. The Proceedings of the annual ACADIA

Annual 2010 Conference, New York.

Kim, J. B., Clayton, M. J., and Yan, W. (2011). Parametric Form-Based Codes: Incorporation of land-use regulations into Building Information Models. Parametricism (SPC). The Proceedings of the ACADIA Regional 2011 Conference, Lincoln, Nebraska.

Kim, J. B., Clayton, M. J., and Yan, W. (2013). Parameterize urban design codes with BIM and Object-Oriented Programming. Open Systems. The Proceedings of the 18th International Conference of the Association of Computer-Aided Architectural Design Research in Asia 2013, Singapore.

Kohr, R. T. (2004). From form to function : an evaluation of the effectiveness and potential of form-based zoning codes (Master thesis). Massachusetts Institute of Technology, Cambridge, Massachusetts.

Lang, J. T. (1994). *Urban Design: The American Experience*. New York: Van Nostrand Reinhold.

Lee, G., Sacks, R., and Eastman, C. M. (2006). Specifying parametric building object behavior (BOB) for a building information modeling system. *Automation in Construction*, 15(6), 758–776.

Ligmann-Zielinska, A. (2008). Exploring normative scenarios of land use development decisions with an agent-based simulation laboratory (Ph.D. dissertation). University of California, Santa Barbara. Santa Barbara, California.

Lunetta, R. S. and Lyon, J. G. (2004). *Remote sensing and GIS accuracy assessment*. Boca Raton, Fl: CRC Press.

McHarg, I. L. (1992). *Design with Nature*. New York: J. Wiley.

Mehta, S. (2006). *Form-based zoning : what place is this code?* (Master thesis). Massachusetts Institute of Technology, Cambridge, Massachusetts.

Melosi, M. V. (1999). The Sanitary City: Urban Infrastructure in America from Colonial Times to the Present. Baltimore: Johns Hopkins University Press.

Nettlefold, J. S. (1914). *Practical Town Planning.* London: St. Catherine Press.

Parolek, D.G., Parolek, K., and Crawford, P. C. (2008). *Form-Based Codes: A Guide for Planners, Urban Designers, Municipalities, and Developers*. Hoboken, N.J: J. Wiley and Sons.

Procedural, Inc. (2011). CityEngine [software]. Available from http://www.procedural.com/cityengine

Punter, J. (1997). The Design Dimension of Planning: Theory, Content, and Best Practice for Design Policies (1st ed.). London: E and FN Spon.

Reynard, P. C. (2002). Public Order and Privilege: Eighteenth Century French Roots of Environmental Regulation. *Technology and Culture*, 43(1), 1–28.

Sacks, R., Eastman, C. M., and Lee, G. (2004). Parametric 3D modeling in building construction with examples from precast concrete. *Automation in Construction*, 13(3), 291–312.

Schumacher, P. (2009). Parametricism: A New Global Style for Architecture and Urban Design. *Architectural Design*, 79(4), 14–23.

SmartCode Central. (2011). SmartCode. Retrieved from http://www.smartcodecentral.org/

Smith, D. K. and Tardif, M. (2009). Building Information Modeling: A Strategic Implementation Guide for Architects, Engineers, Constructors, and Real Estate Asset Managers. Hoboken, N.J: Wiley.

Steiner, F. (2004). Healing the earth: the relevance of Ian McHarg's work for the future. *Philosophy and Geography*, 7(1), 141–149.

Stephenson, B. (2002). The Roots of the New Urbanism: John Nolen's Garden City Ethic. *Journal of Planning History*, 1(2), 99–123.

Sullivan, L. H. (1896). The tall office building artistically considered. Lippincott's monthly magazine, 339, 403-409. Philadelphia: J.B. Lippincott Co.

Talen, E. (2009). Design by the Rules: The Historical Underpinnings of Form-Based Codes. *Journal of the American Planning Association*, 75(2), 144–60.

Talen, E. (2012). *City Rules: How Regulations Affect Urban Form*. Washington DC, USA: Island Press.

Tello, E. R. (1989). Object-oriented programming for artificial intelligence: a guide to tools and system design. Reading, Mass: Addison-Wesley Pub. Co.

The City of Farmers Branch. (2013). Station Area Form-Based Codes. Retrieved from http://www.ci.farmers-branch.tx.us/work/planning/ordinances/station-area-codes

Ungers, O. M. and Vieths, S. (1999). *O. M. Ungers: The Dialectic City*. Milano: Skira.

Walters, D. (2007). Designing Community: Charrettes, Master Plans and Form-Based Codes. Amsterdam: Elsevier/Architectural Press.