PARAMETERIZE URBAN DESIGN CODES WITH BIM AND OBJECT-ORIENTED PROGRAMMING

Jong Bum KIM, Mark J. CLAYTON and Wei YAN
Texas A&M University, College Station, TX, United States
jongbum.kim.coa@gmail.com, mark-clayton@tamu.edu, wyan@tamu.edu

Abstract. There has been a significant need for the new urban design apparatus that carries out performance analyses of the urban models, since the code reform movements focused on the sustainable urban developments. This research explores the use of parametric Building Information Modeling to enable stakeholders to intuitively understand the implications and consequences of urban design codes. We investigate whether key regulations can be captured as algorithms and ontologies in parametric BIM and Object-Oriented Programming. Then we present our prototype of parametric urban models in a BIM platform that explains (1) the extent which urban design code information can be parameterized in BIM and (2) the methods how parametric models can hold code information. The results show that our prototype enables real-time manipulations of code requirements and interactive visualization of code allowances.

Keywords. Parametric modelling; BIM; Object-Oriented Programming; urban design codes.

1. Complex Datasets and Tools in Urban Design

Urban design for community scale developments has been carried out by using complex datasets that are still either digital or analogue. For instance, socio economic and environmental data as well as land-use and zoning information can be obtained from Geographical Information System (GIS) data. Other regulatory requirements are provided in text-based documents. Subdivision plans, created by civil engineers, are distributed in the CAD format generally. Citizen’s inputs from the design charrette are stored in sketches and documents.

In this circumstance, designers juxtapose and synthesize such data to understand what they imply. They create urban models by using a series of computer tools; CAD tools for site plans and three dimensional models; rendering software
for perspectives; spread sheets for feasibility assessments; and energy simulation tools for environmental footprint analyses.

GIS has been used for the repository of natural resources and infrastructures. Urban design regulations are partially stored in GIS data such as zoning and land use maps. Three dimensional design codes also are not included in GIS; in the GIS land use map, a mixed-use zone is displayed with a specific colour code, but further information can be found in the local zoning code document. In addition, inaccurate spatial data of GIS can be used in design process after calibration processes. For instance, when the GIS shape file of the land use layer is exported to the DWG format, inaccurate lines and incomplete polygons can be found. They can be used after considerable post-processing.

This complex use of datasets and tools in urban design can impede interactive feedbacks and rapid design changes. Majority of urban design supportive tools have been developed to scope with interactive visualization and design alternatives, but still there is a significant need for the new urban design platforms for reliable urban modelling, comprehensive data managements, and interactive performance analysis (Barnett, 2011).

2. Modelling of Urban Design Codes

2.1. CODES AND URBAN DESIGN

Urban design codes provide a framework that governs design allowance for various urban design components such as open spaces, public spaces, and buildings. In the United States, recent codes have focused on physical urban form of open space, streets, buildings, and other physical design elements, while conventional codes often focused upon building function and scale (Barnett, 2011; Walters, 2007). However, it has been argued that the contemporary urban design codes employ highly structured code components and prescriptive provisions, and that there are complex interrelationships among such components and provisions (Kim et al., 2011).

2.2. PARAMETRIC MODELLING AND OBJECT-ORIENTED PROGRAMMING IN BIM

BIM has been developed to support the Architectural Engineering Construction (AEC) industry. BIM ties all the building components with imbedded information to create a building product model (Eastman et al., 2011; Sacks et al., 2004). In addition, BIM facilitates communication of design intentions, and coordinates construction processes using BIM’s shared database. Parametric modelling
capabilities in BIM provide mechanisms to map knowledge-based rules into BIM objects, enabling automatic manipulations and semantically-rich model generations (Sacks, Eastman, & Lee 2004). In addition, Application Programming Interface (API) BIM authoring tools provide can support application customization and software development.

This chapter describes a framework of how the parametric BIM modelling can build code components and key provisions, how the API in BIM can establish the interrelationships among them, and how the parametric code models and the applications can communicate.

2.3. BEHAVIORS AND ATTRIBUTES OF BIM OBJECTS

BIM objects are the representation of building components such as walls, roofs, and floors. BIM objects have functionalities that can respond to object manipulations, which can be found in the real-world. These functionalities are behaviours of BIM objects. In BIM objects, attributes are accessed through the parameters. For instance, the wall object in BIM can be moved, rotated, and extended as parametric definitions are changed, but only predefined behaviors can be performed. Most BIM objects represent specific building components and limited behaviors, so they are not appropriate for our parametric code modelling.

User definable objects in BIM allow users to create a BIM object with customized behaviours and attributes that can control geometric representations and properties. The parametric urban code model is created using the user definable objects, so it is required to understand what kinds of behaviours and attributes need to be built and how they can be modelled in BIM.

2.4. MAPPING URBAN DESIGN CODES INTO BIM OBJECTS

Clarifying urban design behaviours and their attributes is the first step of our implementation. For instance, code provisions can secure a minimum open space area by controlling building topologies with maximum widths, depths, and heights as well as building dispositions with setbacks and built-to-lines. In this case, controlling building dimensions and dispositions can be understood as urban design behaviours. In addition, the required values, such as a distance from the property line and building heights, can be defined as attributes. This attributes can be modelled in parameters in BIM objects.

Urban design behaviour can be interpreted as diverse parametric modelling behaviours, which dramatically affects the modelling procedure of BIM objects. For instance, a setback control in the code can push a building façade to the
designated position. On the other hand, an invisible setback line can cut out the building mass outside of the designated position. Furthermore, a void mass can be created to subtract a solid mass outside of the setback lines. Three modelling approaches can present setback requirements, but each one needs to have different behaviour.

3. Experiments

The first implementation step is (1) creating BIM objects that can represent a set of code components and (2) representing the associations among the objects and their parameters using parametric modelling features in BIM. The next step is (3) developing analytical.

In our experiments, we use Revit Architecture as a parametric BIM development interface, Revit API as an application development platform, and C# as an OOP language. With the parametric modelling features in BIM, a BIM object can store information as parameters, the elements of the object can be manipulated according to any parametric values, and relationships among objects can be established. Revit API enables access the model data, generate new elements, edit existing elements, define customized functions and user interfaces, conduct analysis, as well as produce model documents. OOP is an advanced approach in software development, enabling rapid development, high maintainability, and high performance based on a set of rules for operating objects (Cox, 1986; Pinson, 1988).

3.1. DEFINE A PARAMETER SET

Based on a review of existing zoning code, we decided a list of urban components and their parameters that are modelled in the parametric code model. We reviewed existing zoning codes in the United States that outline specific urban form and building design by using (1) Transect and (2) prescriptive provisions. Transect is an urban planning strategy that outlines the hierarchical development scale from suburban to urban cores (Hascic 2006). Each transect zone controls the degree of developments by using particular sets of provisions. Prescriptive zoning codes, in comparison with traditional zoning codes, include more regulatory information of form and geometry (Ben-Joseph 2009). These prescriptive codes intend to reshape urban space and public realm by controlling private developments. They employ built-to-lines to designate the location of the building front facade and siting requirements to designate three dimensional building disposition allowances. In our test case model, following parameters are included.
3.2. CREATE PARAMETRIC CODE MODELS

In the development of the parametric code models, three major urban components are implemented; parcel, building, and parking.

The parcel component holds topology information and regulatory information that can outline open spaces, development capacity, building dispositions, and building envelopes. For instance, the minimum ratio of open space to the property

<table>
<thead>
<tr>
<th>Categories</th>
<th>Parameters</th>
<th>Descriptions</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parcel (Property)</td>
<td>Transect code</td>
<td>Zoning codes from rural to urban</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Open space ratio</td>
<td>The percentage of open space to the property area</td>
<td>%</td>
</tr>
<tr>
<td></td>
<td>Open space area</td>
<td>The lot area excluding structures</td>
<td>ft²</td>
</tr>
<tr>
<td></td>
<td>Property area</td>
<td>The property area</td>
<td>ft²</td>
</tr>
<tr>
<td>Building</td>
<td>Footprint area</td>
<td>The building ground covered area</td>
<td>ft²</td>
</tr>
<tr>
<td></td>
<td>Gross floor area</td>
<td>Total floor area of the building</td>
<td>ft²</td>
</tr>
<tr>
<td></td>
<td>Setback</td>
<td>The distance from the property line to the building envelop</td>
<td>ft.</td>
</tr>
<tr>
<td></td>
<td>Depth</td>
<td>Building depth of the building</td>
<td>ft.</td>
</tr>
<tr>
<td></td>
<td>Building Height (Ground floor)</td>
<td>The first floor height from slab to slab</td>
<td>ft.</td>
</tr>
<tr>
<td></td>
<td>Building Height (Upper floor)</td>
<td>The upper floor height from slab to slab</td>
<td>ft.</td>
</tr>
<tr>
<td>Parking</td>
<td>Number of floors</td>
<td>The floor number of the building</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Required parking number</td>
<td>Total parking numbers according to the building</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Number of street parking</td>
<td>The number of on-street parking spaces</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Available footprint area</td>
<td>The maximum area for the parking structure</td>
<td>ft²</td>
</tr>
<tr>
<td></td>
<td>Planned footprint area</td>
<td>The parking ground covered area</td>
<td>ft²</td>
</tr>
<tr>
<td></td>
<td>Parking space area per car</td>
<td>The parking lot area for one car</td>
<td>ft²</td>
</tr>
<tr>
<td></td>
<td>Number of floors</td>
<td>The floor numbers of the parking structure</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Building height</td>
<td>The height of the parking structure</td>
<td>ft.</td>
</tr>
<tr>
<td></td>
<td>Distance from buildings</td>
<td>The distance from the buildings</td>
<td>ft.</td>
</tr>
<tr>
<td></td>
<td>Distance from property lines</td>
<td>The distance from the property</td>
<td>ft.</td>
</tr>
</tbody>
</table>
area and maximum area of building footprints are described in most code provisions; therefore, the parcel area determines an open space area as well as footprint area of buildings and parking structures. In addition, parcel topology determines some required building lines where the building front façade shall be built or where the main building shall be located. Building setback lines can be generated from the parcel topology.

The building component holds topology information and development capacity information. Prescriptive transect zoning codes specify three dimensional building configurations including building heights, the number of floors, as well as roof and attic heights. The building disposition information of the parcel component should be applied to the building component; required building lines and setback lines of parcel components formulate the building envelope in our modelling process.

The parking component holds parking structure topology information and parking regulatory information. In general, several parking provisions such as the total parking numbers, setbacks, and entrance locations are described in the zoning codes. For instance, the urban design codes (City of Ventura 2007, City of Peoria 2010) designates that a residential building shall provide one parking space for every 1,500 square feet floor. This provision implies that the total parking number can be calculated from the building component. In this circumstance, the parameter values of the parking component can be decided through complex calculations using multiple values from the parcel and the building component.

3.3. MODELLING OF THE ASSOCIATIONS AMONG COMPONENTS AND THEIR PARAMETERS

A set of code components and provisions can be parameterized into a BIM object in diverse ways, but presenting hierarchical associations among the components and their parameters can ease the complexity in the urban model creation.

For instance, a code model can present multiple building and parcel components as shown in the Figure 1. All parameters are included in the same model, so the associations across the multiple components are easily stored in parameter equations. However, the amount of parameter information is too large to be applied for the large scale urban code models having multiple code components. If this approach were applied for the parameter lists in the Table 1, the code model can have more than twenty parameters. If the BIM object had twenty properties, more than four hundred parameters need to be handled. Eventually, this approach would increase the complexity of the BIM object.

In our experiments (Figure 2), we implemented the parametric urban code model having open spaces, buildings, and parking structures by following steps.

- A parking component is hosted by a building component.
A building component is hosted by a parcel component.

A BIM object consists of a parcel component that hosts a building and a parking component. This single object presents one parcel. We call this BIM object as a parametric code object (Figure 2).

An urban model hosts multiple parametric code objects. We call this urban model as a parametric urban code model (Figure 2).
By nesting multiple code objects with the same hierarchy of zoning code components, the parametric urban code model can store parameters in the various but the most relevant depositories; Transects and open space requirements are modelled in the parcel object, the building geometry requirements are imbedded in building objects, and parking parameters are modelled in parking objects.

3.4. DEVELOP ANALYTICAL APPLICATIONS

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 stored in parameter equations because a BIM object cannot automatically and intelligently recognize the parameters within other BIM object.

An example is Floor Area Ratio (F.A.R.), a ratio of the total floor area of the buildings to the property area. To obtain this value from the parametric urban code model, two values are needed; gross floor area and property area. The gross floor area can be read from the building object and the property area can be read from the parcel object. 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 object, and it can be transferred to the building object. However, the building object in the model cannot read parameters from other objects.

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

Following code examples show the Revit API that can (1) read parameters of the entire model, (2) calculate F.A.R. values using given parameters, and (3) display information to the user interface.

The first step is selecting a parcel component in the Revit user interface to read its name as shown in the Figure 3. The collected values will be used for locating a correspondent building component and a parking component from the entire urban code model. To do so, we retrieve the 'name' value of the 'Parameters' class of the parcel component. After that, other internal variables of the code name and the parcel name are defined. A loop construct of foreach is used to apply the same process to the entire parcels.

Next step in Figure 4 is collecting required parameter values for the density calculation from the selections. The property area, transect zoning codes, and the parcel name are collected and the data types are converted.
Figure 5 shows an entry point to access a building component. We named the building component as the codeModel. The required parameters are stored in the 'Parameters' class of the codeModel, so we define an internal parameter set with this class.

The area of total building floors can be obtained from the building component parameters. The parameter name value is Gross Floor Area, and the value is stored as a double type. We store the value into the internal variable of TFA as an integer. Using two variables of the parcel area and the total floor area, we can calculate F.A.R. that can measure the development density (Figure 6). The last step is to

```csharp
/// 8) Calculate Floor Area Ratio
foreach (FamilyInstance parcelModel in parcelFamilyInstances)
{
    /// string which will be passed to ModelInfoViewer
    if (parcelModel == null) continue;
    StringBuilder sb = new StringBuilder();
    /// Parameter "Volume" can be obtained from ParameterSet!!!
    ParameterSet lotparameters = parcelModel.Parameters;
    string modelName = parcelModel.Name.ToString();
    string parcelName = modelName.Substring(0, 4);

    // Internal variables to calculate F.A.R.
    lotArea = parcelModel.get_Parameter("Area").AsDouble();
    int parcelArea = (int)lotArea;
    lotCode = parcelModel.get_Parameter("Code").AsString();
    lotName = parcelModel.get_Parameter("Name").AsString();
    string fileName = lotName + lotCode;

    foreach (FamilyInstance codeModel in codeFamilyInstances)
    {
        ParameterSet parameters = codeModel.Parameters;
        string faNames = codeModel.Name.ToString();

        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 3. A code block for locating a BIM object set from the parametric urban code model.

            Figure 4. A code block for collecting variables for the density calculation.

            Figure 5. A code block for entering a building component.

            Figure 6. A code block for calculating density using internal variables.

            Figure 5 shows an entry point to access a building component. We named the building component as the codeModel. The required parameters are stored in the 'Parameters' class of the codeModel, so we define an internal parameter set with this class.

            The area of total building floors can be obtained from the building component parameters. The parameter name value is Gross Floor Area, and the value is stored as a double type. We store the value into the internal variable of TFA as an integer. Using two variables of the parcel area and the total floor area, we can calculate F.A.R. that can measure the development density (Figure 6). The last step is to
store internal variables into a parameter list as a series of strings, which will be presented in the viewer in Figure 2.

4. Conclusions

In this research, we implemented the parametric urban design code model using BIM and OOP. It could visualize key provisions of urban design codes and the structure of code components. The applications of BIM API could facilitate interactive communications among multiple code components, which could assess the information within multiple code components and generate new values using such component information. 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.

Provisions governing quantitative aspects of urban design, such as urban form and capacity can be modelled in our prototype, which do not present qualitative aspects of urban design codes. Further research needs to focus on how our prototype can support such aspects of urban design codes.

References


Cox, B. J.: 1986, Object-oriented programming: an evolutionary approach, Addison-Wesley, Reading, MA.


