Building Your Own Urban Tool Kit

Utilizing parametric BIM components as smart early design tools for large-scale urban planning

Caner Dolas¹, Andreas Dieckmann², Peter Russell³
Chair for Computer Aided Architectural Design, RWTH Aachen University, Germany
http://caad.arch.rwth-aachen.de
¹dolas@caad.arch.rwth-aachen.de, ²dieckmann@caad.arch.rwth-aachen.de, ³russell@caad.arch.rwth-aachen.de

Abstract. The paper describes the development of a set of smart BIM components to facilitate and accelerate the creation of large-scale urban models in the early design phase in a BIM software environment. The components leverage the analytical, parametric and modelling capabilities of the BIM environment to support adaptive parameter-driven building geometry, patterning of different building types, early numerical and graphical design evaluation, various simulation methods and the exploration of design alternatives. The toolset consists of the most common building shapes, but can be extended with additional shapes and their respective area and volumetric calculations when necessary. The rapid large-scale deployment of the components has been achieved by diverting existing tools from their intended use.

Keywords. BIM; urban planning; early design; rule-based design; parametric design.

PROJECT CONTEXT: BIM VS. GIS.

Building Information Modelling (BIM) is quickly becoming the de-facto standard in the computer aided design and documentation of buildings, albeit with varying adoption rates in different world regions (McGraw-Hill Construction, 2010 & 2012). Data structures in BIM applications can be described as semantic, parametric and component-centric (Eastman et al., 2008). BIM applications utilize the Industry Foundation Classes (IFC) file format that was first released in 1996 (Bazjanac and Crawley, 1997) for the exchange of semantic data models.

For the urban scale, similar efforts to create a semantic data model have been made with CityGML (Gröger and Plümer, 2012). Although, compared to IFC, CityGML is a relatively young data format (Kolbe et al., 2005), it is supported by a growing number of commercial software products [1]. At present, however, none of these are BIM applications (and are thus located outside of the “comfort zone” of architects), but the issue of interoperability between building and city models has become a hot topic of research in recent years (Nagel and Häfele, 2007; Isikdag and Zlatanova, 2009; El-Mekawy, 2010; De Laat and Van Berlo, 2011). Therefore it stands to reason that in the foreseeable future BIM applications will gain the capability to author semantic urban models.

The parametric and analytical capabilities of BIM applications have proven to be somewhat useful for urban design in the past (Miller et al., 2009).
and research has been conducted towards the implementation of zoning requirements in building information models (Donath and Lobos, 2006; Kim et al., 2011). However, with increasing project scale, the mere process of placing a large number of diverse elements, altering their attributes and exploring design alternatives has left much to be desired with regard to speed and usability.

**REQUIREMENTS FOR A SMART TOOLBOX**

The focus of the project was to create a toolbox of smart components that could be used as generic building masses inside a BIM application for large-scale urban planning projects. Building models in urban design projects are usually made up of a limited number of building archetypes, yet each building instance has to accommodate the geometric conditions and zoning requirements of its respective parcel as well as the overall design intent. Additionally, there is a strong need for evaluation, especially in the early design phase, in order to facilitate informed decision-making. Hence, the following requirements were set for the components:

1. A large number of components can be distributed rapidly in a given spatial framework, making it possible to create a large-scale urban model in a fairly limited amount of time.
2. The components can adapt to varying lot sizes and geometries.
3. The components allow for the rule-based parametric generation of building forms according to zoning requirements.
4. The components can accommodate different terrain conditions, i.e. they work on level and sloped terrain surfaces.
5. It is possible to automatically or at least semi-automatically place components in patterns (such as ABAB etc.) to allow for variations in the design.
6. The components can instantly report most if not all essential numerical information relevant in a typical urban planning scenario such as areas, volumes etc. but also, if possible, some statistical data on costing or environmental impact (e.g. CO\(_2\) footprint).

In turn, the requirements for the host application itself were defined as follows:

1. Allow for the creation, distribution and maintenance of components according to the requirements listed above.
2. Generate tabular reports of component attributes.
3. Allow for visual filtering based on component attributes.
4. Facilitate the creation and comparison of design alternatives.
5. Possibly even provide additional analysis tools.

Based on the above requirements, Autodesk’s Revit platform [2] was chosen as host application. Not only does it meet all the requirements, but its conceptual modelling application Vasari [3] also includes analysis tools for environmental factors like sun and wind.

**DESIGNING FOR RAPID DISTRIBUTION**

Repetition and variation are common concepts in architecture. They can be easily identified in building elements such as curtain walls, staircases, railings, structural systems etc. BIM applications generally provide dedicated tools for these types of building elements. The same concepts of repetition and variation apply to urban planning as well, perhaps with a special emphasis on the adaptability of buildings to the geometric conditions of their respective parcels. However, there are no dedicated tools for distributing a large number of building masses in a typical BIM application. Therefore, the approach was to divert tools readily available in the chosen application from their intended use.

With the 2010 version of Revit, Autodesk introduced a new conceptual modelling environment that was intended for the modelling of building masses [4]. The potentials of this modelling environment were described by Miller et al. (2009), but the workflow outlined by them involved the manual modelling of each building (or at least manual changes to placed building instances). The 2010 version did, however, come with another functionality...
with a lot of potential regarding the adaptability of a large number of objects to varying geometric conditions: Mass surfaces could be rationalized by using the “divide surface” functionality and subsequently be populated with “pattern-based curtain panels”. Revit 2011 saw the introduction of the “adaptive components” functionality: placement point based components that can adapt to varying spatial conditions [5]. Lastly, with the 2013 version came the “repeat and divide” workflow that can be used to create more complex arrays of objects (Dieckmann and Kron, 2012) and facilitate the large-scale distribution of reactive components (“reactors”) as described by Woodbury (2010).

Surely none of these functionalities were designed with large-scale urban planning in mind – most of them are typically used for the creation of curtain wall systems and other building elements – but they can be “abused”. In the context of the project, the aforementioned tools are used as follows:

1. The footprints of city blocks are created as mass surfaces (Figure 1a).
2. These mass surfaces can then be subdivided into lots using the divide surface functionality, creating a grid within the city block. The grid can either be generated automatically (Figure 1b) using a layout algorithm (e.g. number of subdivisions in U/V direction) or manually (Figure 1c) by drawing a number of lines to generate the subdivisions.

The actual toolbox consists of several types of building masses created as pattern-based elements and adaptive components that can be hosted on and rapidly distributed across divided surfaces. Depending on the desired outcome, two separate modelling strategies can be applied for populating the grid with the building masses:

1. For a simple pattern, the divided surface can be assigned a pattern-based component (Figure 2a), essentially distributing instances of the same building block across the entire grid of a block. Exceptions can be defined by selecting individual instances and manually switching their type or altering their instance properties (Figure 2b).
2. More complex patterns of several alternating building types can be created as one or two dimensional arrays by employing the repeat and divide workflow (Figure 2c). In addition, this workflow allows for the rapid deployment of context-aware adaptive components that can, for instance, react to the proximity of other objects in the model (Figure 2d). A common application for this method would be the increase of density towards certain zones in the urban model (see below).
COMPONENT OVERVIEW

For the purpose of surpassing a mere proof-of-concept stage, component types were developed for most commonly found building shapes: I-shaped, L-shaped, U-shaped, O-shaped and solitaire. The lot and building block components are organized in a nested object structure (Figure 3). The lot component, a pattern-based element, is intended for:

1. Placement on and distribution across the city block’s grid.
2. User input. Depending on the component design, the input can consist of different types of rules and constraints such as building dimensions, setback, plot area ratio (PAR), floor-to-floor height, usage type, building orientation etc.
3. Evaluation of lot geometry (dimensions and angles, where applicable).
4. Communication of user input and lot geometry to the nested building block component.
5. Calculation of the required numerical data needed for design evaluation (e.g. building footprint, building volume, cubic index etc.).

Nested inside the lot component are one or several instances of building block components. These adaptive components are linked to their parent component by parametric relationships. As they are created as what is referred to in Revit as “non-shared” components, they are completely absorbed by their parent component and can neither be selected nor scheduled as separate elements in the project environment. They mainly consist of “dumb” geometry and only perform the following tasks:

1. Evaluation of the input received from the parent component.
2. Generation and positioning of building geometry in the context of the lot geometry based on the received inputs.

In the case of more complex design intent such as gradually increasing the building density towards a subregion within the planning area, lot components can be nested in another context-aware adaptive component that is able to track its proximity to said subregion and drive these parametric constraints in the building block components, as described by Dieckmann and Kron (2012) for curtain wall panels.

Lot Component Anatomy

The lot component is created as a pattern-based element, a component that is based on a number of placement points. As the lot component has to adapt to varying geometric conditions set by the geometry of the city blocks, it needs to be aware of its own shape and size, i.e. the lengths of its edges and the angles between those edges. In Revit, such properties can be measured by using so-called reporting parameters that report the varying dimensions for each placed instance of a pattern-based component. While the components may be placed on sloped surfaces, the dimensions need to be measured in top projection in order to be used for the calculation of areas and lengths later on.
4a). This is done by hosting all the dimensions on the horizontal work plane of the first placement point.

The geometry of a pattern-based component by default inherits the orientation of its host, i.e. the divided surface of the city block. That means that vertical elements created in the lot component would rather orient themselves according to the surface normals of the city block than vertically at their point of placement. By changing the orientation mode of the placement points the lot component geometry can however be forced into a strictly vertical orientation. The placement point location can then be projected upwards by means of vertical rays. On sloped lots, the building may have to be moved up or down so as not to be fully or partly immersed in the terrain. This can be achieved by creating a horizontal datum between the aforementioned rays (Figure 4b) that can be moved by manipulating a parameter that controls the vertical offset of the datum.

The horizontal datum serves as the placement plane for the building component itself. It is subdivided into nine zones by projecting the street offset for all four sides of the lot onto the datum (Figure 4c). These offsets can be controlled by the user through four parameters. In case the street offsets of opposing sides of the lot overlap, the user inputs will be substituted by a “safe” value that is automatically calculated.

The four intersection points of the street offsets form the location for the placement points of the building component (Figure 4d) and also mark the vertices of the central zone that forms the basis for the building footprint calculations (see below). Once a building component is placed here, its type can be controlled by a parameter, making it easy to change the orientation of the component (front, right, back and left side of the lot) as well as the building shape (I, L, U, O). This also allows for the subsequent creation and substitution of other building shapes essentially making it a modular system. Additionally, all the parameters that control the building shape (building depth for all sides of the lot and building height) are also passed to the subcomponent. As stated above, the building subcomponents merely consist of the building geometry driven by the lot component parameters and thus warrant no further description.

For the purpose of calculating the building footprint and related data like floor space and building volume, the central zone is again subdivided into nine zones, this time by using the building depths for the four sides. Again, the depth for each side is user-controlled with a safeguard against overlaps as described above for the street offsets. The footprint of each building type can now be calculated as the sum of some of the zone areas (Figure 5), depending on the selected building type, e.g. the footprint of the O-shaped building would be the sum of all zones except for the central zone. The zone areas themselves are calculated on the basis of the reporting parameters (see above) using Heron’s formula and the law of cosines. Subsequently, all other data necessary for evaluation such as cubic index, floor area ratio or site occupancy index can be derived from the building footprint, the number of floors, the floor height and the site area. In Revit, custom component parameters can not be scheduled or annotated in the project environment by default.
Thus, in order to have the data readily available in the project for evaluation, they need to be declared as so-called “shared” parameters making them available globally (in the component itself as well as in the project).

**Component Variations**

The lot component can be used as a template to create further variations. They can either be different building types than the four types described above, more complex parametric components that utilize the lot component as a subcomponent or a combination of both.

The solitaire component (Figure 6a), for instance, makes use of the spatial and parametric framework of the lot component. However, it needs neither the street offset grid nor the majority of parametric relationships that aid with the area calculations for the standard building types (I, L, U, O). Instead, it contains a center point for the free-standing building geometry that can be moved parametrically in U and V direction on the lot surface. The building geometry that is hosted on the point in turn has a rotation parameter to allow for flexible alignment of the building mass.

A reactor component (Figure 6b) as described above can use either the solitaire component or the standard lot component as nested subcomponent. It is basically an adaptive component that sets up rules for the behaviour of its subcomponent. It has one or several additional placement points that act as sensing devices. By hosting these additional placement points on certain fixed points in the project and measuring their distance from each placed...
instance of the reactor component, the components gain spatial awareness. This information can then be used to control the geometric properties of each placed subcomponent, e.g. the number of storeys.

WORKING WITH THE TOOLKIT

The typical workflow has been, at least in part, described above already: The city blocks are created as mass surfaces and subsequently subdivided into lots. Depending on design intent, several distribution methods (uniform, uniform with exceptions, patterned and reactive/parametric) are available (Figure 1). The component type(s) assigned to a block, a lot or a pattern can be changed and their instance properties can be modified. The shapes of the mass surfaces themselves and the number of their respective subdivisions can also be modified at any time. Moreover, several out-of-the-box functionalities like design options (managing different design alternatives) and phasing (managing the temporal properties of elements, i.e. differentiating between existing and new building blocks) can be utilized to structure and control the design.

The main reason for using a BIM environment for urban design, however, is the ability to create information-rich content and leverage that information to evaluate the design. All the numerical data produced by the placed components can be easily scheduled. Each lot component contains a flag parameter that facilitates the creation of a schedule that only displays the lot components placed in the project and ignores all other site components available in the model. The schedules can utilize conditional formatting to highlight lots that do not meet certain requirements like, for instance, a cubic index that exceeds a certain limit (Figure 7a).

A schedule is, however, just one way of looking at information. The same information can also be visualized in isometric, perspective or plan views, displaying the information in a spatial context. In Revit, model views can be reformatted with so-called view filters. By means of a few view filters a perspective view of the project can be colour-coded according to value ranges of any given parameter like, for instance, the cubic index of each lot, with different colours for different value ranges (Figure 7b).

Often, the building type has a significant influence on the measurable characteristics of a building. For instance, the energy use of a building depends quite heavily on the activity within that building. There are some statistical resources available for that kind of information, like the Buildings Energy Data Book by the U.S. Department of Energy [6]. However, for the purpose of this paper, the authors have focussed on costing. In a lot of countries, there are statistical data available on the building costs for various building types. For the german market, this data is made available by the BKI Baukosteninformationszentrum (2013). In Revit, external data can be inserted in the form of so-called key schedules, either by inputting it manually or by using third-party applications [7] to import it from Excel. A row of values from a key schedule can be assigned to a placed component by means of a key parameter.
After that, a costing schedule can easily be created that contains parameters that, for instance, calculate the building cost on the basis of the building volume and the cost per cubic meter specified in the key schedule for a particular building type.

Once the schedules and filtered views have been set up properly, the project file can be used to create a project template for future urban planning projects. This way, the information will be readily available as soon as the designers start placing the first lot components – they could even model the project in a filtered view for direct visual feedback.

**DISCUSSION**

The presented method facilitates a quick, albeit makeshift, workflow to create early design models for large-scale urban planning projects in a BIM application. All relevant numerical data is generated on-the-fly by the components themselves as they are placed in the context of the site. The design can therefore be immediately evaluated – either numerically or graphically – making it easy to explore different design alternatives. Additionally, the chosen host application has the capability of performing environmental analyses for the impact of sun and wind on the design. The components, of course, still have some limitations, e.g. a useful functionality would be to be able to assign more than one usage type to a building, perhaps per storey.

For the sake of interoperability, a sensible next step would be to reach the capability to export the model to CityGML format. Previous research on the subject of marrying IFC and CityGML quoted in this paper has focussed on the conversion of entire building models to several levels of details (LOD) in CityGML. In this specific case, a conversion of single elements (generic models) in the building model to LOD 1 or 2 CityGML building entities would do the trick.

**REFERENCES**


Nagel, C and Häfele, KH 2007, Generierung von 3D-Stadt-


