Conceptual Design of High-rises with Parametric Methods

Victor Gane¹, John Haymaker²
Stanford University, USA
¹vgane@stanford.edu, ²haymaker@stanford.edu

This paper describes the use of parametric methods in generating conceptual designs of high-rise buildings. We first assess current conceptual design practice at a leading AE firm and illustrate the challenge of satisfying a complex set of requirements with the tools currently used by the AEC industry. Few design options are generally developed and analyzed; better design solutions are most likely being overlooked. Parametric tools can potentially help address this problem by allowing designers to formalize and generate solution spaces that can be explored. But few case studies exist to document the construction and impact of these models. We present such a case study. We describe the variables, constraints, components, and rules in the model. We discuss the costs and benefits, and conclude with recommendations for expanding the use of parametric methods.

Keywords: Conceptual; rule-based; parametric design.

Introduction

Conceptual design is a challenging part of the design process as it requires reconciling the design with a complex set of goals and constraints such as program requirements, construction cost, environmental performance, and more abstract notions such as aesthetics and usability. When faced with such complexity, designers are taught to generate and test a large number of options. However, faced with time and budget constraints, Architecture Engineering and Construction (AEC) project teams today often generate and test relatively few options for a design problem. Analyses of these options are limited, and tend to favour space programming and aesthetics over other criteria. Two possible causes of these problems lie in the tools the AEC industry is using.

First, these tools were designed to support generating single, static solutions, they do not efficiently support exploring and managing many potential design alternatives. Second, these tools do not yet integrate well with many analysis tools, therefore discouraging multidisciplinary analysis. Potentially better building solutions are neglected by the current inability of design teams to rapidly generate and analyse a wider range of design options. This paper deals principally with the first issue: generating design options.

Parametric modelling promises to revolutionize the way we design buildings. To design parametrically means to design a system that sets up a design space which can be explored through the variations of the parameters. In this paper we explore the use of parametric methodology as a potential solution...
medium for high-rise design and test the extent to which it helps address the above described limitations. We first describe our observations of current high-rise design practice in the United States. We then introduce parametric methodology. Next, we describe a case study in conceptual high-rise building design where parametric design is used to model and manage changes. We conclude with a discussion of the strengths and weaknesses of this method.

**Benchmark current practice**

When designing high-rises, architects rely on multiple sets of rules (Stiny, 1980, Alexander et al., 1977). Among these are reusable building typology rules that help generate the optimal lease-spans (residential – 30’, office – 40’), or meet minimum building efficiency requirements (65% → net sellable area = gross area - core area). There are also explicit, project-dependent high-level rules such as the local building codes (i.e. setback from property line) or stakeholders’ requirements (i.e. min. floor plate area) used by designers as general guidelines for generating the building’s volume. Designers rationalize these rules in more implicit, low-level geometric rules that determine geometric element dimensions and relationships. The concurrent implementation of such rules in traditional CAD is challenging and time consuming, especially when multiple goals are tracked simultaneously. Limited, rudimentary analyses are performed by engineers at this point because of schedule constraints and incompatibility of architect-generated design information for use in discipline-specific tools. In a two-week timeframe designers normally develop as few as 1-3 options. Figure 1 describes this process. Explicit analyses are normally performed after the conceptual design phase, which often leads to major changes to the original concept, or unsatisfactory design results.

Our research investigates whether parametric tools can help formalize the complex rules that describe a high-rise and how might this improve the current design process.

**Parametric design-concepts**

Originating in the field of mechanical design, parametric methods are gaining popularity in the AEC industry. A few practitioners (Whitehead et al., 2003, Shelden et al., 2002, Hesselgren et al., 2006) and academics (Burry, 2003, Woodbury, 2006, Killian, 2006) are actively implementing parametric and/or performance-based design on AEC projects but industry wide adoption is hindered by the lack of knowledge about how to apply parametric thinking to different building types and among multiple disciplines. More formal case studies and metrics are needed to help establish the benefits and drawbacks of parametric methods in the AEC industry. In this paper, we describe our parametric methodology around six main concepts: variables, constraints, dependencies, components, PowerCopies, and rules.

**Variables** are the primary drivers of geometric variations. We distinguish between two types of variables: independent and dependent. An independent variable is a user-defined numeric input whose value can actively be controlled and changed (i.e. triangle height etc.) while the dependent variable is the output whose value changes as a result (i.e. triangle area). Variables can also be global or local depending

---

*Figure 1*  
Narrative of the current SOM conceptual design process
on how these have been attributed to geometric elements. For example, by attributing a variable to the radii of all columns in a building one would establish a global variable, since modifying its value will propagate globally to all the columns. In contrast, a local variable will always affect only a single geometric element to which it is attributed.

**Constraints** help delineate the range of variations that a parametric model can sustain. The extent of the range and the exact outcome of each geometric variation will be determined by the type of constraints used in the geometry definition process. We differentiate between two types of constraints: *dimensional* and *geometric*. Dimensional constraints are equivalent to the above discussed variables, in which an attributed numeric value acts as a constraint until it is modified. Dimensional constraints are essential in defining the geometry of a design concept. For example one might define an arc by constraining its radius, and length. Such constraints establish a dependency of the geometric elements on the variable(s) that defines them.

Geometric constraints help determine how the geometric components relate to each other, establishing another level of dependency. For example, one might use geometric constraints to tangentially inter-constrain two arcs such that the final definition of one arc depends on the other and vice versa. Geometric and dimensional constraints work in tandem to enable the dynamic character of parametric models. A summary of constraints and the types of geometric elements to which these apply is given in fig. 2. The symbolic representation of each constraint is derived from CATIA – one of our research tools.

A **Component** is an assembly of geometric elements defined by variables and constraints (i.e. four coincidentally and perpendicularly constrained lines with constrained length and a width variable produce a column profile).

A **PowerCopy** (PoC) is a set of building components that are grouped and intended to be used in a context. A PoC definition process may include specifying input variables that allow this PoC to be instantiated according to the context into which it is pasted. For example, one might define a column extrusion along a centerline with a height and inclination angle; the latter two become input variables and when the PoC is pasted into a floor with a different height and inclination angle, it uses the specification of new variables representing that specific condition leading to an automated update of the PoC geometry.

The concept of PowerCopies suggests the need of deconstructing a parametric model into assemblies...

**Figure 2**
Dimensional and Geometric constraints and types of geometric elements to which these apply

<table>
<thead>
<tr>
<th>Constraint Type</th>
<th>Representations</th>
<th>Applies To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensional</td>
<td><img src="image" alt="constraint representations" /></td>
<td>Lines, Ellipses, Lines &amp; Ellipses</td>
</tr>
<tr>
<td>Geometric</td>
<td><img src="image" alt="constraint representations" /></td>
<td>Arches, Circles, Ellipses, Circles, Arches, Ellipses, All Geometric Elements</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constraint Type</th>
<th>Representations</th>
<th>Applies To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed</td>
<td><img src="image" alt="constraint representations" /></td>
<td>All Geometric Elements</td>
</tr>
<tr>
<td>Horizontal</td>
<td><img src="image" alt="constraint representations" /></td>
<td>Lines, Ellipses, Lines &amp; Ellipses</td>
</tr>
<tr>
<td>Vertical</td>
<td><img src="image" alt="constraint representations" /></td>
<td>Lines, Ellipses, Lines &amp; Ellipses</td>
</tr>
<tr>
<td>Coincidence</td>
<td><img src="image" alt="constraint representations" /></td>
<td>Points &amp; Any Other Geometric Elements, Arcs, Circles, Ellipses</td>
</tr>
<tr>
<td>Concentricity</td>
<td><img src="image" alt="constraint representations" /></td>
<td>Lines, B-Spline Control Point Tangency Direction, Lines, B-Spline Control Point Tangency Direction</td>
</tr>
<tr>
<td>Perpendicularity</td>
<td><img src="image" alt="constraint representations" /></td>
<td>Lines, B-Spline Control Point Tangency Direction, All Geometric Elements</td>
</tr>
<tr>
<td>Symmetry</td>
<td><img src="image" alt="constraint representations" /></td>
<td>All Geometric Elements &amp; A Symmetry Line</td>
</tr>
</tbody>
</table>
of “intelligent” components later instantiated into as many copies as required and constrained to building components that form the parametric skeleton (i.e. multiple instantiations of the column profile PoC populated on footprint dependent control points).

A Rule is used to construct a component from inputs. Input types range from numeric and geometric variables to constraints and other components.

A design Theme is a collection of rules determining the geometric topology and the global behavior of the parametric model.

**Infinity Tower case study**

We use these definitions to discuss the application of parametric methodology to the Infinity Tower – an all residential project designed by SOM in 2005-06 and currently under construction in Dubai (fig. 3).

The design process started with an assessment of the site configuration, a set of stringent client-defined program requirements (i.e. particular unit mix within specific area ranges, overall building efficiency $>70\%$) and the city codes (i.e. offset from property line – 1.2m, tower placement $\geq20$m to the adjacent building on its right) (fig. 4). Based on the above-mentioned constraints the optimal footprint emerged as a narrow, elongated configuration (fig. 4). This led to the choice of the twisting tower design theme, because this move exposed the best units in the top third of the tower to the best views of the ocean and the city skyline.

The first author developed a parametric model in CATIA for improved management of the complex nature of the geometry and the ability to systematically explore a series of design alternatives. The model first defines the building’s footprint, and core, then extrudes and twists the buildings floor plates and places columns based on a global twist.
parameter. The model next constructs the window wall, and architectural fins based on this underlying structure. We describe below and graphically in fig. 7, the rules used to construct each component in this model.

**Construct ground footprint rule**

This rule constructs a description in plan of the building perimeter at the ground plane. The input types range from ones based on high-level reasoning (i.e. design theme leading to decisions about the design's macro features) to low level reasoning (i.e. dimensional (var) and geometric constraints (con) for implementing the design theme geometrically).

Should the choice of geometric topology be opposite (i.e. curvilinear vs. rectilinear), the nature of this rule would change and a different set of constraints would be used.

**Construct core & circulation corridor rule**

Given a Ground Footprint and the twisting tower design theme, this rule constructs 2D circles to describe the footprint of the building core and the interior corridor. Considering the rotation of each floor at a...
constant angle, a circular core was anticipated as the most optimal configuration for achieving uniformity in the unit layouts.

Should the choice of the theme be different (i.e. flaring vs. twisting tower), the core rule would change, leading to a different set of constraints and variables being used (i.e. rectangular vs. circular core may become optimal).

**Construct Column Footprints**

This next rule takes the ground footprint and establishes the locations for all of the columns on the ground plane. It first creates construction points used for constraining column profiles to the ground footprint. Then it creates a 2D rectangular column profile dependent on one control point.

Together, these rules establish the building at the ground plane. Similar rules were then used to construct descriptions of the building’s envelope, columns, beams, slabs, walls, windows and shading fins. Following is a brief description of these rules:

- **Construct Footprint PoC** – assembles footprint and column profiles into a PoC that is instantiated twice to create the mid and top tower footprints. The ground, mid and top footprints as well as the column profiles are controlled by global variables used to initially construct the ground footprint and the column profile PoC. *Tower twist* variable is the only exception. It requires formulaic adjustment for the mid footprint PoC instantiation to always be half the value of the global variable ($tower\ twist = tower\ twist/2$) and for the ground footprint to always maintain a zero value by subtraction ($tower\ twist = tower\ twist - tower\ twist$).

- **Construct Tower envelope** – creates the tower envelope as a surface by lofting the ground, mid and top footprints.

- **Construct Column extrusions** – creates 3D extrusions from column profiles at ground, mid and top footprints with tangency dependency on the tower envelope.

- **Construct Single Floor Slabs** – creates two slabs from intersecting the envelope with two planes controlled by the *floor height* variable.

- **Construct Single Floor Columns** – slices column extrusions with the floor slab definition planes and extracts an output variable calculating *surface area*.

<table>
<thead>
<tr>
<th>Input</th>
<th>Reasoning</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground footprint (comp)</td>
<td>The rule first requires column definition control points to be coincidentally constrained to the Ground footprint component, to establish a dependency relationship. The distance between control points is managed by the Column spacing variable. The rule uses Column Length and Column Width variables to geometrically define the column profile. The Coincidence constraint maintains the dependency of the column’s midpoint (exterior side) to a column definition control point, while the Perpendicularity constraint maintains that relationship between column’s long and short sides. A second Perpendicularity constraint establishes the relationship between column centerline and ground footprint. Since multiple identical copies of columns are needed, the rule requires building a PowerCopy that will be instantiated multiple times. Column Length, Column width and Column spacing become PoC definition input variables to help establish a dependency of instantiated copies to those variables.</td>
<td>Column footprint (comp)</td>
</tr>
<tr>
<td>Coincidence (con)</td>
<td></td>
<td>Column PoC Instantiated PoCs</td>
</tr>
<tr>
<td>Column spacing (var)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Column length (var)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Column width (var)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coincidence (con)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perpendicularity (con)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perpendicularity (con)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 3**

*Construct column footprints*
• **Construct Spandrel Beams** – creates a spandrel beam outer footprint, offset from the floor slab perimeter and controlled by the *offset from slab* variable. The inner footprint is then offset from the outer footprint and its position controlled by the *width* variable. The rule manages the beam extrusion height with the *height* variable. An output variable calculating the beam’s *volume* is then extracted.

• **Construct Glazing Single Bay** – creates within a single window bay (space between columns) a glazing position footprint dependent on the slab perimeter through the *offset from slab* variable. The rule then divides the glazing vertically into 3 equal panels by first creating a dependent variable (*window length = column spacing – column width*) that feeds into another dependent variable (*vertical division = window length / 3*). It then divides the window panels horizontally into vision glass and shadow box. The division is generated with a plane controlled by the *spandrel beam height* variable, which establishes an explicit dependency to the spandrel beam. Two output variables are then extracted calculating the vision glass and shadow box areas.

• **Construct Fins** – creates 3 fin panels within a single bay by using the *vertical division* variable and then extracts an output variable calculating *surface area*.
Construct Window Bay PoC – creates an assembly out of the single bay glazing and fins, controlled formulaically by the selected PoC input variables. The window bay PoC is then instantiated multiple times to fill in all the bays of a single floor. Because of the twisting geometry, each column has a different inclination angle, making the use of a window bay power copy highly effective.

The model supported efficient exploration of design options within the range of the chosen theme. Since the tower had four unique floor types, rationalizing the exterior wall as a single floor allowed rapid generation of new floors by adjusting the value of the floor height variable (fig. 5). In conjunction with structural engineers the model was used to iteratively refine the design in response to the evolving structural solution (i.e. modified columns spacing, column size, spandrel beam height and width). The area calculations, as well as the glass and fin solutions were automated and each floor type exported for assembly in traditional CAD tools.

The completed model took one person a week to construct. It became an essential part of the design process with the client’s visit to the architect’s office. In two days the design team was able to assemble and visually present three options based on the client’s comments (45 deg, 60 deg, 90 deg rotation and glass repositioned to exterior face for maximizing sellable area) (fig. 6).

In retrospect, as constructed the model can respond to relatively simple changes in the design theme (i.e. from twisting tower to vertically extruded by modifying the tower twist value to zero or from twisting to leaning by modifying the distance to origin variable in the mid and top footprints or from uniform twist to tapered twist by modifying tower side length variables at the mid and top footprints, etc). However, the model will not support investigation of alternate, more complex design themes (i.e. blob tower) without more pre-rationalization in constructing the parametric skeleton to include a wider range of the solution space – a challenging task in the usually fast paced conceptual design process.
Many of the underlying rules for the current twisting tower theme will not apply for a “blob tower” and much of the parametric skeleton and dependent geometry will need to be reconstructed with a modified set of rules. Furthermore, with simple parametric techniques, the current model cannot easily deal with modifying the number of bays or vertical division of glazing panels. To achieve this level of flexibility, a parametric model must be complemented by custom reasoning (i.e. scripting to allow the use of loops).

**Conclusions**

This paper presented a documented case study of a parametric model constructed for one high rise building. More case studies and theory are needed to understand the ways in which parametric thinking affects the design process. The case study illustrates that parametric tools can enable architects to shift from creators of single designs to designers of systems of inputs and outputs that generate design spaces. Such tools provide a foundation to formalize design spaces in terms of explicit theme-dependent rules and manage information in a unified environment. However, designers need to understand and reformulate the design process in these terms, which implies new ways of thinking and communicating about design. Parametric models can provide important geometric flexibility and support an iterative design refinement process. However, the choice of rules used to construct the model will limit its usability to a specific range, often making extreme design variations unfeasible.

With the ability to rapidly generate design options comes the need to analyze them all. To enable multidisciplinary integration, parametric tools can be used to construct and manage layers of geometric information optimized for discipline specific analysis tools (i.e. stick models for structural analysis, meshed geometry for CFD or Incident Solar Radiation analysis etc). As long as the dependency of each layer is established to the primary parametric skeleton (i.e. in Infinity Tower case to the ground, mid and top footprints and the envelope), the geometry update process can be automated. Additional rules establishing the data schema and the information flow between parametric and multidisciplinary tools, and methods to manage and optimize these processes also need to be developed.

**References**


