Parametric Design Process of a Complex Building In Practice Using Programmed Code As Master Model

Kat Park and Nicholas Holt
Parametric Design Process of a Complex Building In Practice Using Programmed Code As Master Model
Kat Park and Nicholas Holt

Abstract
Parameter based design explorations inevitably require a unified master model that represents the current design state, where each parameter being explored is essentially a critical sub-case of this master model. Throughout the constantly changing design state, it is beneficial to maintain a master model that is flexible and adaptive. This paper describes the design process of a complex building whose master model documented the design logic through implementation of software code. This process is illustrated by the case study of Lotte Super Tower (Seoul, Korea) from the beginning of schematic design to end of construction document phase. By maintaining the master model as a platform-free software code, in contrast to platform-dependent methods, the case study illuminates the advantages of documenting the generative logic behind design variations in a way that allows greater flexibility and a higher level of alignment with design intent.
1. Introduction

Integrating parametric models into the process of design requires a formalization of the generative logic and a systematic way of evolving this logic in concert with design changes. A successful master model must, therefore, keep track of the various parameters being explored. The critical sub-cases that these parameters represent might be as simple as studying the effects of a numerical value on the building’s form or a much more complex set of values that explore the building’s response to factors such as environmental conditions. In the more complex scenarios, it might be more efficient for the explorations to take place outside of the master model; nevertheless, whatever design decisions might have been made need to be fed back into the master model in order to propagate its influence on the overall design, as well as manage the relationships among the various sub-cases. Parametric representation of a design is a constantly evolving task, since most complex design problems in practice are continually reacting to addition and deletion of inputs at multiple scales and levels of complexity. Setting up a parametric model requires defining the major parameters in the initial stage, but this initial set of parameters continues to grow and change as the project develops into different stages. Parametric modeling platforms existing today offer assistance in defining the parameters and their associative relationships; however, it is inevitable that using their predefined set of tools will make the representation dependent on the specific platform. The notion that availability of a tool invites use is well accepted in design. Different modeling platforms demand the end-user to have differing levels of understanding and rigor in the process of defining the parameters and their relationships. Rigor obtained through the constraints opposed by a software platform is not only limited and often biased, but it is also dependent on the form defined by the platform. As design process is inherently “parametric,” it is important that designers use an open platform that is most appropriate for building the relationships as they arise.

The main roles that should be fulfilled by the unified master model are:

- offer a platform for the iterative-evaluative loop for design options
- capture and document knowledge of other related disciplines affecting the design
- provide a platform for negotiating optimization of competing disciplines.

This paper demonstrates a new method of managing a complex parametric design system where programmed code captures the understanding of the inherent generative logic as well as how the system might be constructed or fabricated in an efficient and flexible way, while fulfilling the main roles of a master model mentioned above.
2. Existing Approaches to Master Data Models

It is commonly agreed that all design tools have particular affordances and biases, and they implicitly embody particular assumptions and values. In general, a designer’s toolkit represents a provisional equilibrium of capability and demand.[1] In this section, we provide an overview of design tools commonly used in practice to aid in the design of projects similar in scale and complexity to Lotte Super Tower, seek to illuminate the assumptions and values embodied in each, and assess how well each meets the criteria of a successful unified master model as outlined above.

2.1. Master Data Model Using Non-parametric Platforms

Non-parametric methods of representing a design are the most widely used digital design tool in today’s practice, and thus provide a baseline against which to assess the value of more advanced techniques. Setting up the logic for a parametric model requires initial investment of time and effort, which always need to be evaluated against the projected lifespan of the model. A quick conceptual study might sometimes call for a model without the need to document the logic or associative relationships, or the initial resources might simply not be available.

A 3D Rhinoceros model of a form (without the use of Grasshopper plug-in) is an example of non-parametric modeling platform. In this platform, one method of exploring or documenting design intent is to use geometrical elements and geometrical operation tools offered by Rhino. This method is perhaps most susceptible to the design being influenced by the tool. While a complete account of modeling by direct manipulation of the geometry suggests a design process that is beyond the scope of this paper, we may speculate that a systematic approach to form-generation is possible in this media by using points, construction lines, etc, to leave a trace of the logic applied in creating the form. In other words, the subassembly of parts used to create the desired form is deliberately saved by the operator.

Figure 1. Construction figures are saved in order to document the design generation explicitly
This approach, the manual parametric exploration, leaves a history of the generative logic, but the set of data representing this logic is rarely recoverable and difficult to decipher. More often than not, different design options created in this manner require similar amount of data and work-hours for each additional option. If this method persists into the later stages of design, the representation needs to be updated to the new design state by repeating the modeling effort each time. While this representation might capture the knowledge of a partnering discipline (for instance, duct size coordination with mechanical consultants requires the floor-to-floor height of the model to change), the knowledge is indirectly embedded in the model, and thus hidden.

2.2. Master Data Model Using BIM Platforms

Parametric modeling have matured in recent years well beyond the state of art when they first appeared on the professional architectural design stage, both in terms of computational power and well-resolved user interface. BIM systems like Autodesk Revit and Dessau’s Digital Project provide a platform to represent and manage 3D building components and systems, capable of integrating the various members in design and construction phase such as structural engineers, MEP engineers, and fabricators. By applying object-based parametric modeling, controlled manipulation of properties and inheritance, these platforms organize the building data using the general object-oriented computation concepts. These platforms run behind an interface that is slightly closer to how the software themselves operate than their predecessors, and require that the user learn the structure of their language. In exchange for this greater investment on the part of the user, the database becomes more computer-interpretable, which carries with it a wide range of advantages: exporting quantities of materials and assorted other metadata about the components and systems, carrying two-way links to building specifications, automatically translating the design information into analysis applications, integrating energy, structural and user simulations to inform the design, and many more.[2]

One of the key benefits of such BIM system is to create a single central model that can be explored from the view of multiple disciplines of the design. Integrated coordination with structure or MEP arm of design is possible when the “architectural model” is compared to “structural model” through clash-detection; but much of this expectation-realignment can happen at a conceptual level if there were an efficient way to communicate the design logic or the structural logic. With Revit or DP master models, the architectural model can be coordinated with the structural model only when the structural engineers have put together a structural model. In initial form-finding stages, the current set of interfaces in Revit imposes more constraints than invite use. More often than not, formal explorations are executed elsewhere (typically a more flexible environment such as Rhino)
and imported into Revit in order to document or append more construction-relevant data. For this reason, it is difficult to discuss Revit’s current state as a parametric design platform, as it is used more successfully as a project delivery platform.

Digital Project’s engineering-caliber geometry engine provides the designer a toolkit for constructing complex 3D surfaces and solids. They provide abstractions and interfaces to help define the design problem, interfaces to organize and manage the parameters and their relationships with each other. The specification tree lists all the geometry and features but this organization method is rigid which binds the designer to think of the parameters in a particular, Digital Project-centric way. Moreover, unless the associations and parametric relationships have been developed properly to anticipate certain changes, not only is it difficult to alter them, it is difficult to maintain any parts that depend on a relationship that might need to change. Especially at earlier stages of design, such constraints imposed by the platform are not desirable. Scripting or creating macros within DP can relieve some of the rigidity but the user is still tied to the platform-specific geometrical primitives. For example, a designer might speed up the generation of surfaces through a script, but the script must refer to the DP definition of a surface, not a mathematical definition of a surface. The parametric modeling toolset inherent in these BIM systems provide a snapshot of the artifact being designed; but they cannot capture the intent or the progression of the design logic through various iterations and changes, either due to immaturity of the toolset or its rigidity.

2.3. Master Data Model Using Parametric Design Platforms

McNeel’s Grasshopper is much more flexible as a parametric design tool. The designer can explore multiple ways of generating one form or element,
which can be easily saved as a cluster of modules, often within the same Grasshopper file. This allows the designer to save the generative logic in an efficient, organized manner. Nevertheless, the designer is still tied to a very particular user interface, and consequently, only explore within its boundaries under its indirect design influences. For instance, one quickly runs into the way data structures are inherently handled within all the modules and its large influence in many of the tasks. Because the data structures are required to be set up in a particular way, users often resort to creating workaround solutions rather than build a parametric relationship or association that is intuitive to them. Creating one’s own modules have recently become fairly easy, which have added greater flexibility. More importantly, Grasshopper has not yet been able to handle the amount of data or complexity of a project described in this paper.

3. Case Study of Lotte Super Tower

3.1. Project Background

The geometry of 555m tall Lotte Super Tower begins with a constant transformation from a square base to a circle top. This idea is conceptually structural as well as architectural: transformation in order to shed wind vortices which occur in unchanging form, taper for efficient mass distribution of mixed-use program which require varying lease spans. The geometrical challenge of transforming a square into a circle was resolved by creating triangular facets on the building.

Figure 3. Lotte Super Tower
3.2. Design Variation Exploration

It is important to note that the master model described here did not grow out of a desire to extend the capabilities of AUTOCAD platform as a design tool, but rather the need for a platform for the mathematical development of the geometrical design moves evolving in the tower. The operations in the programmed code did not use AUTOCAD specific functions, but AUTOLISP was used for its ease of handling lists of data. The formulas and generative logic were described in the code with AUTOLISP; the parameter values were kept in a separate configuration file that the software read in as part of its initialization process; and different forms of outputs were generated for different needs. The series of configuration files, each associated with a specific version of the logic file, made it possible to keep an organized record of design status as constant changes were being applied.

Figure 4 shows a simple example of the descriptive logic that was used to provide iterative-evaluative loop for design exploration in the initial phase of the project: the overall dimension or the number of facet modules for proportional evaluation, as well as functional evaluation such as floor plate dimension requirements, enclosed area, etc.

![Figure 4. Initial parameter set for overall tower dimensions, subdivision modules, and their relationships.](image)

\[
\begin{align*}
W &= \text{Square Base Width} \\
r &= \text{Circle Radius} \\
M &= \text{Module Division} \\
\theta &= (\alpha - \beta) / M \\
X(A_{\alpha}) &= r \cos \alpha \\
Y(A_{\alpha}) &= r \sin \alpha \\
X(B_{\alpha}) &= r \cos \beta \\
Y(B_{\alpha}) &= r \sin \beta \\
X(A_{\beta}) &= r \cos (\alpha + \theta) \\
Y(A_{\beta}) &= r \sin (\alpha + \theta)
\end{align*}
\]

Figure 5 shows the simple parameter set and their relationships from Figure 4 translated into a function named BuildData in AUTOLISP code. As mentioned previously, the specific values of the parameters are stored in a separate configurations file in order to keep an organized history of the values at different design stages, often accompanied by the reasons for their change from the previous version. The BuildData function abstracts the idea of a line at the top and bottom of the tower, not a circle or a rectangle. This lower-level of abstraction allowed the flexibility of changing the top figure...
into a square (which was one of the numerous design explorations the project went through) or a circle. This kind of flexibility can often be difficult to achieve when the platform restricts the operator to use its specific set of user interface tools to create parametric relationships.

3.3. Knowledge Capture of Partnering Disciplines

Documenting the generative logic into software code also allowed coordination of design intent with structural engineers to begin at a conceptual level, as well as capture their knowledge into the master model. The optimal angle for the diagrid structure (diagonal columns than span the subdivided arcs across the discretized elevation points) was determined mathematically as a function of the load each member was required to carry at their individual locations. Since the angles cannot change mid-member, the results from the theoretical solution was reinterpreted with discrete values pertaining to the dimension of the diagrid, which is shown in dark blue lines in Figure 6. Translating this type of information into software code allowed the communication between the design team and the structural engineers to remain at a level that provided accuracy and efficiency unachievable by 3D models. The values of $z_n$ elevations specified at right in Figure 6 indicates where these structural diagrid discretizations occur in the tower. These values were primarily determined by programmatic and architectural requirements, which also fluctuated throughout the design phases. The exact nature of the lines which connect the top arc and bottom line, as well as the logic of exactly how these
discretizations occur along the lines were all design explorations that affected architecture as well as structure. These ideas were translated into code and passed back and forth with the structural engineers sometimes as code, sometimes as pseudo-code. When parameter values changed, coordination was as easy as communicating a new set of values, since both disciplines had already agreed on the parameter logic and their relationships through code. This type of platform-independent communication between the disciplines facilitated a generative - as opposed to a strictly analytical - structural design consultancy, and provided significant flexibility in contrast to the platforms typically used by architects and structural engineers.

3.4. Documentation of Critical Sub-Cases

The master model served as a central repository of various iterations and results obtained from specific analysis. This section describes two such cases: installation feasibility and fabrication analysis of exterior wall detail and exterior wall panelization strategies.

Figure 6. Left: Engaging Structural Engineer’s theoretical solution to the optimal diagrid angles. Right: The final values of \( z_n \) elevations documented at the end of Construction Documents phase.
Exterior Wall Fabrication Analysis

Rule-based logic documentation was critical in exploring the building’s exterior wall design, since the geometry of the tower’s exterior wall was determined by offsetting the structure’s centroidal wireframe in a systematic manner. After exploring numerous offset methods, we found that the size of the diagrid nodes was the governing factor in determining the specific offset dimensions. In other words, the final offset dimension values specified in Figure 7 represents the minimum distance necessary for the exterior wall to clear the structural diagrid nodes at each elevation. The new set of horizontals derived from this method defined the external boundary of the exterior wall.

After the horizontal offsets were determined, the diagonal centroid which locates the line of the diagrid structure was offset as well. Defining this relationship between structure and curtain wall was a complex task, since it spanned from a problem of geometrical concept all the way to understanding fabrication and installation requirements.

The following diagrams show one example of the offset strategy which was explored in order to determine a structure-to-exterior-wall relationship that satisfied multiple conditions.
The variations of offset strategies were designed with rule-based logic in the master model, after which the results were exported to be visualized in renderings and drawings to ensure design intent. At the same time, the relationship was tested for fabrication feasibility through a more comprehensive torque analysis in Digital Project. This required the master model to output the exterior wall geometry and the structural geometry into a set of coordinate data that could be read into Digital Project.
More than ten different geometrical relationships between structure and exterior wall were explored during this process, and since all the relationships were documented as software code, generating the result of a particular offset method as coordinate data (delimited text file) was natural. Maintaining the logic as programmed code allowed flexibility to produce various outputs each catered for different needs, eliminating the problem of competing file formats prevalent in traditional BIM approach. Figure 10 depicts one particular example of the coordinate data generated for the triangular facets which describe the surface of the exterior wall. This coordinate data was imported into a Digital Project model to measure the level of torque or clearance in order to ensure ease of fabrication and installation. Modeling the multiple variations of the relationship between the structure and exterior wall with a parametric platform like Digital Project would not only have taken much longer due to its heaviness, but also would have required a specification tree size similar to that of multiple (and separate) instances of the model. This is a clear advantage of maintaining a master model as a central repository of design logic, both documenting design intention and capturing the recommendations of fabrication and construction consultants.

**Exterior Wall Panelization**

The exterior wall of the tower is divided into three distinct sections, each...
with different set of conditions and characteristics that required a different panelization logic:

1. At podium, the exterior wall is composed of approximately 45m tall cable net wall. Therefore, panelization strategy prioritizes continuation of structure cables across adjacent triangular facets (tri-surfaces).

2. Most of the office and hotel zone of the tower is curtain wall. Anticipating interior partitions, perpendicular mullion fabrication takes priority over mullion alignment across adjacent tri-surfaces. In a few cases, the anchor point of the vertical grid shifts to straddle the midpoint of the triangle, rather than coincide with the midpoint. This was a simple solution to maximize mullion alignment across adjacent tri-surfaces.

3. The exterior wall at the observation deck and restaurant zone is free from any interior wall module requirement; therefore, visual transparency is prioritized. To minimize view obstruction, cable net structure is defined by a diagonal grid, producing diamond or triangle panels.

Due to the taper and transformation in the tower form, each triangular facet (tri-surface) of the exterior surface resulted in a unique geometry, aside from the reflective symmetry across the center of the overall surface. Therefore, it was necessary to handle panelization of each surface individually. The logic of how each surface should be panelized was explored and documented in the software code; and depending on the location of the surface, one of final three strategies mentioned above were applied to the
Once a panelization strategy was chosen, a more quantitative analysis of the panelization was performed. For this purpose, the software produced a delimited text file of coordinates, dimensions, and angles for every panel in the curtain wall. This initial panel data was quantified to assess how many panels were regular (rectangle) and how many were irregular.

As shown at left in Figure 12, it was necessary to quantify the small irregular panels adjacent to the expression zone that could not be fabricated as individual panels. After consulting curtain wall engineers and other various parties involved in manufacturing and installing a curtain wall, a procedural algorithm that described a strategy for combining the small irregular panels into its larger neighboring panel was developed. The rule-based logic in the algorithm implemented the level of certainty offered by the consultants. Variables such as the maximum acute angle or the minimum width of a glass piece depended on the manufacturer, which became parameterized inputs in the algorithm.

Running the algorithm to create panel combinations according to manufacturer’s criteria produced a data set for each panel in the curtain wall as well as quantify the types of fabrication necessary for the entire wall, allowing preliminary cost estimation, etc. An optimization routine as comprehensive as this may only be performed algorithmically and only implemented within a procedural environment. The methodology of keeping a code-based master model offers the designer a more seamless
optimization environment in comparison to other methods which might allow script routines. The hierarchical nature of BIM or 3D parametric modeling tools can be an obstruction when trying to integrate script routines into specific locations in the model hierarchy. In the case of procedures that need to return to particular places to reevaluate and essentially disrupt the hierarchy, a software code can achieve this with little effort.

3.5. Facilitating Negotiation of Competing Optimizations

After initial panelization of the exterior wall, the pure geometrical results were evaluated against construction conditions. Various manufacturers and other disciplines related to exterior wall, as well as architects with years of construction experience, combined their expertise and wisdom regarding the available construction tolerances and the accuracy that is practically required by a panel dimension. Taking these factors into account, the original panelization procedure ran an additional layer of optimization that grouped certain panel dimensions into an average dimension in order to standardize panel types for fabrication purposes. This type of optimization was conducted for both the curtain wall and cable net wall at the observation deck. In the case of optimizing the cable net wall panel dimensions, it was necessary for the changes in the dimension to be tested for structural integrity of the cable net concurrently. Multiple disciplines competing to optimize for its own best solution is natural in a design process. Due to the inherent ability of a programmed code to document such generative logic, the master model was able to facilitate the negotiation of trade-off optimizations of multiple disciplines, as well as document the compromises, abandoned variations and final results of the optimization.

Figure 13. Left: Visual representation of the exterior wall panelization. Middle: The rule-based logic for combination documented in code. Right: Visual model of this logic applied to the original panelization method.
4. Conclusion

Parametric thinking in design may be conceived of as the process of abstracting a specific task into its operation and inputs. For many designers, scripting is the first step to explicitly defining and documenting the design logic. As the Lotte Super Tower project demonstrates, when approached as a loosely organized network of scripts, a master data model in the form of programmed code can provide significant flexibility, efficiency, and coordination capability, especially in the context of complex and prolonged design problems in practice. In large complex projects, the malleable code can respond to constantly growing and evolving changes in the project much faster and more easily than other comparable software platforms.

Acknowledgements

The author would like to thank all members of Lotte Super Tower project team.
References


Kat Park and Nicholas Holt
Skidmore, Owings & Merrill, LLP
United States

Email: kat@alum.mit.edu; nicholas.holt@som.com