CONTROLING DESIGN VARIATIONS

DESIGNING A SEMANTIC CONTROLER FOR A GENERATIVE SYSTEM

Pedro Engel[1]
[1] *Universidade Federal do Rio de Janeiro*
[1] *pedroengel@fau.ufrj.br*

This article will describe the recent steps in the development of a computational generative system based on the selection and combination of ordinary architectural elements. Built as a Grasshopper definition, the system was conceived to generate designs of architectural façades and to produce models, physical and digital, for didactic use. More specifically, The paper will address the conception of controlling devices, that is, the parts of the computational system that govern design variations. This process involved two complementary actions: first, the definition of a clear organizational logic, where elements can be represented as a data structure that encompasses classes, sub-classes, sets, libraries and attributes; secondly, the establishment of means to operate the variations through the use of filters and heuristics based on visual patterns, allowing varying degrees of automation and user control. It will be argued that such organizational model paves the way to increase the number of design possibilities in the future and to and provide means to integrate of architectural criteria into the generation process. This research has received the support of CNPq.

Keywords: *Algorithm, Parametric Design, Architectural Design, Teaching , Physical Model*

INTRODUCTION

Governing design generation is a key aspect of computational systems that aspire any degree of automation. In developing means to govern a generative process, two issues are vital. First, the question of how much is humanly controlled and how much is to be run by a pre-programed algorithm. Secondly, the question of the means or parameters elected to inform design generation. If human control is allowed, should design options be selected directly or, instead, may one define a set of criteria, as in a brief, and let the system generate solutions? Or else, if system generate designs automatically, how can we establish means to maximize desirable solutions and rule out unacceptable ones?

This article aims to discuss these issues by recounting the experiment of conceiving the control devices for a computational system named Supergrid, which generates façade design by selecting and combining ordinary architectural elements . The ex-

periment entailed two complementary efforts: first, establishing a clear organizational logic for the elements and formal variations in design; and secondly, defining the controlling devices themselves.

In the first part, as a means to contextualize the work and the issues at stake, the system will be described in terms of its origin and purposes. In the second part, the paper will describe how the organizational logic has been structured in order to allow the governing of design solutions. The third part will discuss the means employed to govern design variation and present four methods actually implemented: (1) full human control; (2) full random selection; (3) filtered random selection and (4) pattern oriented filtered random. Finally, potentialities envisioned in terms of pedagogical potential and architectural reasoning will be discussed.

SUPERGRID, COMBINATORY SYSTEM

Supergrid is a computational system devised to produce models of façades - physical and digital ¬- to be used as didactic tools for teaching in a second semester design studio in the Faculty of Architecture and Urbanism of the Federal University of Rio de Janeiro. The studio focuses on fundamental architectural issues and on developing design reasoning by letting students face semi-structured architectural problems. All exercises follow a kit-of-parts logic (Love, 2003), where formal variations are restrained to a limited set of known architectural elements distributed in a modular grid, as described by Engel and Eskinazi (2016). The formal language chosen derives, on one side, from mid-century Brazilian modern erudite architecture and, on the other, from the infill architecture of ordinary buildings that are pervasive in cities that rapidly grew during last century, portrayed by Aureli (2014) as a realization of the logic envisioned by Le Corbusier for the Dom-Ino system. The exercise which relates more directly to the Supergrid is called "Module / Façade as Interface", which consists of combining basic building parts (beams, pillars, slabs, walls, windows, doors, shades, filters) to design the front and back façade of a public building facing a square. The aim is to put students in contact with an extensive set of fundamental architectural issues pertaining to the façade: visual permeability and privacy, natural lighting and solar intake, access and use, ventilation and shading, visual composition and meaning. One key aspect is to differentiate the structural frame from the closing planes and shading devices. Besides, it allows students to experiment with the surprisingly wide range of formal variations permitted by such a restricted set of elements and, specially, to exercise the syntactical and parametric reasoning embedded in the orthogonal modular system set by the brief.

Supergrid was conceived to produce models that are used in the studio as didactic tools. They function as a kind of catalog, presenting a wide range of possible syntactic variations whose implications can be discussed in the light of the issues mentioned above. With the help of these models, teachers can advocate design criteria by relating formal configurations to their potential performance in actual building. Hence, Supergrid is, so far, an apparatus to discuss and think about architecture, not to design it.

Figure 1
Physical model generated by the computational system, built in the first phase of development.

In addition, it has been used as an opportunity to make research by design and to explore computational issues related to the creation of a generative system based on the selection, combination and transformation of predefined elements. The first experiments, described in an article (Engel (2018) included creating a Grasshopper definition that em-

bedded a kit-of-parts logic. Although the work employed some of the principles present in the literature about shape grammar (Stiny, 1977; Duarte, 2005; Beirão, Duarte and Stouffs, 2009; Eloy and Duarte, 2014; Garcia, 2016), it used neither its characteristic method of rule applications, nor the capability to recognize emergent shapes (Stiny, 1993; Stouffs, 2019). The system was structured using combinatory logic, where formal variations resulted from the selection and grouping of elements (or parameters) already present in predefined libraries. In the first phase the Grasshopper definition allowed to automatically generate line drawings to produce 1:50 physical models via laser cutting. A collection of joints was carefully devised to allow easy assembly and neat presentation.

The control of design variations in this phase was, however, either openly random or humanly operated. This imposed limitations the expansion of the models and made compositional arrangements subject to the operator's formal preconceptions. Moreover, the system still did not prevent "unacceptable" designs to happen (for example, nothing prevented a balcony without a parapet to be designed), thus design generated by automatic random methods frequently failed to meet basic architectural criteria.

The second phase advances towards enabling semi-automated designs generation and to incorporate design criteria into the system. This required providing the system with an algorithm that governs the selection and combination of elements. As described by Kowalsky (1979), an algorithm is composed of a logic component - an organizational structure that logically links all elements in the system - and a control component - which governs the tasks performed (STOUFFS and HOU, 2018). In Supergrid, all elements and varying parameters are part of a logic structure that can be represented as a database composed of libraries contained elements with attributes. Such logic structure allowed the controlling devices to be conceived as filters that determine which elements are elective for random selection depending on their attributes. This paves the way, it will be argued, to both expand the system and integrate architectural design criteria as means of controlling the generative process, allowing the system to become semantically coherent.

ALGORITHM, LOGIC COMPONENT

In Supergrid elements are organized in classes and sub-classes, combined to form sets and assigned attributes related to their formal characteristics. However as a starting point, the distribution of all elements and their variations use are guided by a tridimensional grid defined by a few basic parameters: module, number of spans in x, y and z directions, and their dimensions (which are multiples of the module). The façade plane is divided into "cells", which are numbered sequentially, beginning from the bottom left and following up in the same column. Equally the lines between two neighboring cells in the façade, called interstices, are numbered sequentially. These numbers operate as addresses, allowing the system to distribute the design variations in the façade.

One way to explain how elements and their variations are logically related in the system is by clarify the vocabulary employed.

Module: Parameter that function as a unit for establishing dimensions. Must be a multiple of 15cm (which is the measure of a 3mm mdf board in 1:50 scale).

Cell: Space between grid lines in the façade plane, having the lower left corner as a reference point. In a building it correspond to the free space between structural elements.

Interstice: Lines dividing two neighboring cells in the façade plane. Interstices can be either vertical or horizontal. In a building they correspond to the outer limit of pillars (vertical) and floor slabs (horizontal).

Address: Number of a given cell or interstice.

Element: General concept that refers equally to a specific part of a building or to the kind this part belongs. Since it is insufficiently precise to help ordering the system, a more precise terminology had to be adopted.

Class: Term used to categorize distinct kinds of architectural elements, such as 'pillars', 'floor slabs', 'beams', 'walls', 'parapets' and so forth. Each cell or interstice can, potentially, contains an elements of every class.

Sub-class: Portions of elements that exist only inside one specific class. For example, the class 'parapet' contains sub-classes 'parapet.front', 'parapet.leftside' and 'parapet.rightside', referring to the three parts that may compose a balcony parapet.

Type: Term used to designate the different shapes the elements of a certain class can assume within a given class, forming a "library" of types. For example, the class 'wall' has a library of 25 types, which differ according to the shape of their openings. Some classes have only one type, as the class 'pillar', in which elements are all shaped the same way.

Library: Collection of types of a given class. Is ordered as a database structure, containing types and with their attributes.

Attribute: Defines the characteristics of a specific type in a library. Attributes are always expressed by a value or code and may derive from a geometric property or be assigned to a type arbitrarily. For example, all types in class 'wall' have attributes referring the shape of their openings, such as 'o_percent' (referring to the percentage of the area of the opening in relation to the total surface of the closing plane), or 'o_verticality' (referring to the coefficient that relates height and width of a given opening). Libraries and attributes form a database built via Python scripting using the GhPython component inside Grasshopper. This internal database is a convenient solution for it allows to retrieve information from the very shape of the types present in the library.

Set: Special type of element, formed by the combination of variations of elements pertaining to different classes, forming a set-class. For example, sets in the set-class 'façade.depth' consist of combinations of 'walls' and 'floors' assuming in positions. Sets can also have attributes of their own.

Set-class: Term used to categorize sets.

Set-library: Collection of sets that belong to a specific set-class.

Instance: Specific occurrence of an element in the model as a result from the generation process.

Variation: Define the designs options allowed by the system. Variations can mainly be of three kinds: "toggle" (on/off), "position" and "type". They are named in the format 'class_variation', as in 'wall_position', 'wall_type', 'parapet.front_toggle' or 'parapet.front_type'. Variations are assigned independently for each cell or interstice in the façade. However, as we will see, the control component can condition the occurrence of a variation depending on other variations either assigned to neighboring cells or to elements of other classes in the same cell.

Variation list: Is a list of values that will command variations to be applied in each cell or interstice.

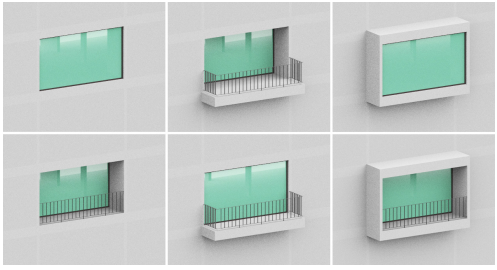

Figure 2
Sets present in the set-library of the 'façade.depth' set-class. All sets result from the combination of one 'wall_position' and one 'floor_position' variation. Parapets, lateral dividers and covering (corresponding to the 'floor_position' of the cell above) are automatically inserted via mandatory conditioning.

As mentioned, there are three main kinds of variations supported. The first, "toggle", defines if a certain class of element exists or not in a given cell, as in a true or false statement. The second, "position", requires to assign a value that will tell how many modules an elements, or its edge, will move inwards or outwards in relation to the façade plane. The third kind, "type", consist in choosing which type - which shape - among those available in a given library, an element will assume when inserted in a cell.

Also, variations can be designated via the selection of sets. As mentioned, a "set class" is a special kind of class that consists in combining variations of at least two distinct classes. The sum of all combinations possible forms a set-library, conjoining sets with their own attributes. The attributes that re-

sult from the combinations of two variation tend to be more relevant architecturally. For example, the depth of a balcony as an attribute of the sets in the set-class 'façade.depth'. It results from the combination of two variations, 'wall_position' and 'floor_position'. Only the relationship between this two values can determine whether there is a balcony and what is its depth. Set classes allow to choose combinations, instead of choosing variations independently. Still, set classes exist only to facilitate the governing of variations in the generation process. In the formation of the model itself, the two variations ('wall_position' and 'floor_position') are assigned as separately.

Concluding, distribution and combination of variations is precisely how the Supegrid performs design generation. To understand how this is operated is of major importance to comprehend how the control component is conceived. Variations are commanded by lists that assign values to each individual cell or interstice in the façade (value indexed 5 in a list corresponds to cell or interstice addressed as 5, for instance). So far, the variations performed the model are defined by more that sixteen lists that operate independently to form the digital model. It is through the composition of all variation lists that the system operates the different design possibilities. Thus, governing the composition of lists is equal to governing the process of design generation.

ALGORITHM, CONTROL COMPONENT

The control component of the algorithm embedded in Supergrid was designed to allow different combinations of automation and user control. Also, it offers distinct ways for the user to operate variations, ranging from direct interference in the design of every cell to the indirect control via filtering devices. In this section, four methods experimented to govern design generation will de described. Before, however, some explanation regarding the conditioning of variations will be given.

Automatically conditioning variations

In Supergrid, conditioning is the term used to define mandatory restraint of variations, that is, variations not subject to election in the generation process. It consist basically in overriding choice made by the user, or in automatic generation modes, because it produced some combination considered invalid by the program. Conditioning is relevant part of the control component, since it prevents the system to generate designs considered unacceptable. Variations may be conditioned by the position of the cell they occupy (ground cells are not the same as top floor cells) or by other variations happening in neighboring cells or, still, by variations of another class happening in the same cell.

An example in Supergrid is when a wall advances outwards. Both the floor above and the floor bellow must go together with it. To make wall and floor position intrinsically dependent, the variation 'floor_position' variations is conditioned by the 'wall_position' variation in the same cell and in the cell bellow. As a result of this rule, the system can guarantee that whenever a wall is pushed outwards, the system automatically brings out the floor above it. This automatic control device relies on embedded codes, usually a simple Python script with an "if / then" statement. This logic makes it necessary to establish priorities, defining which variation will be changed when an unacceptable combination is found.

Methods for governing design generation

In the following section, the experiments in controlling design variation will be described and, consecutively, discussed in terms of their logic, limitations and potentialities. Four modes of governing design variations were experimented: (1) full human control; (2) full random selection; (3) filtered random selection and (4) pattern oriented filtered random selection.

"Full human Control", as the name indicates, is a mode where variations rely completely on human choice. In this mode values have to be assigned to every individual cell independently (although some-

times, variations selected manually may be overridden by mandatory conditioning). One of the limitations of this method is the interface, which requires a large number of inputs (which limits the possibility of expanding the model) and a knowledge of the relation of cause an effect between input and design variation. Human governing also means renouncing the possibility of seeing unexpected designs as an outcome of the generation process. Design intention is completely delegated to the user and, therefore, put forward his or hers preconception and inventiveness regarding visual composition and design ideas. On the other hand, this mode - and precisely because it relies on the operator's intentions - shows great pedagogical potential by turning the kit-of-parts into a game-like experience that favors design experimentation within the systems formal language.

"Full random selection" is on the opposite side of the control-automation scale. Here all electable variations are subject to random selection with no limitations besides the mandatory conditioning described above. The results are always unexpected and tend to appear unordered. Interestingly, they remind of the urban landscape in areas of Brazilian cities where there is nearly no participation of professional architects in designing the built environment. On the other hand, these results may seem too unordered for the eye of trained architects. Also, the full random selection gathers disparate solutions, suggesting the inexistence of consistent criteria throughout

Figure 3
Example of façades generated with the four modes of selection, from top to bottom: full human control; full random selection; filtered random selection and pattern-oriented filtered random selection.

the façade. Depending on the intention, it seems desirable to endow the system with means to apply some kind of order or consistent logic to the generation process.

"Filtered random selection" is a mode that allows the user to limit the number of variations that will be subject to random selection. User operates by defining several different filters to elective the variations 'wall_type', 'façade_depth' and 'shades'. Moreover, different filtering values may apply for cells on ground level, intermediary floors and top floors. The filters are based on the attributes of a given library of types and are also programmed via Python scripting. Hence, this method relies heavily on the database generated within the logic component of the algorithm. Most filters work by defining minimum or and maximum values accepted for a given attribute. The program filters out all types whose value for this attribute fails to fall within the established range. Alternatively, filters may be defined as a range of possibilities. For example, if user states that the system a façade may allow façade extrusions, but not subtractions, the random selection will leave out all 'façade.depth' types where 'wall_position' recedes from the façade plane.

So far, filters operate by relating directly to the attributes of a given library of types. Although understandable by the developers, this option may seem too abstract for an average user to operate. A path yet to be experimented is to congregate filtering values in predefined scenarios, which may be suitable to a given design situation. For example, a scenario labeled as "south façade" would favor large glazed windows (if context is south hemisphere and bellow the tropics), while if labeled "west façade", it would provide maximum shading elements, small windows or variations that result in deep façade compositions.

Another issue verified in this mode of control is that filters may or may not be used intentionally to promote visual ordering of the design generated. Along the developing process, intentions to generate visually coherent designs led to the formulation of a fourth mode of control, where user has greater control over the visual distribution of the filtered selections over the façade.

"Pattern-oriented selection" applies the same logic as the method described above, but instead of distributing filtered variations among ground, body and top floor, the system allows the distribution of variations to be guided by a visual pattern whose configuration is either assigned by user or randomly formed. The pattern is defined as a matrix filled with values ranging from 0 to 5, where rows and columns refer to the cells of a small portion of the building to be replicated along the façade. When a value is repeated, the variations assigned to the first occurrence of that cell, is also be repeated in the building. When patterns are assigned manually, generation process is subject to visual oriented criteria, offering, once again, pedagogical potential to discuss visual thinking along with the performance oriented criteria favored by the filtering devices.

DISCUSSION

The experiment recounted in this article is part of an effort to conduct investigation by design. While describing the creation of control devices for a computational system developed in Grashhopper, it hoped to touch in relevant issues related to the problem of governing design generation in systems that aim some degree of automation. A few observations regarding such issues may be of value at this point.

Firstly, the experience made evident that a well ordered logic component is vital to provide means to developing powerful and flexible governing devices. Moreover, since the logic component is structured as a database, it offers the appealing capability of supporting the addition of objects to the system, which may allow to increase the richness of its design language it operates.

As far as the tension between automation and user control in the governing component, this issue brings forth concerns expressed by Mario Carpo (2014) regarding the responsibility of designer knowledge in the emergence of artificial intelligence and automated systems: how much is to be left for

the designers to design? The answer to this question may depend on the intensions and purposes of the system. Thinking about Supergrid in the context of design education, the idea of flexible system that allows to switch from human control to automatic generation is appealing. Human control seems to offer rich pedagogical potential and an experimental tool. Hence, probable further development will be on the way of providing a more human-friendly interface for the system. On he other hand, while the full random control appeared to offer lack of order or logic, the installation of customizable filters in the algorithm seemed a powerful way to provide indirect control over the design generation process, calling for a reframing of the design thinking, from visual to parametric.

Finally, the possibility of constructing means to translate the numeric operators of the filters in terms of architectural criteria raises fundamental issues abouth the nature of design activity. While it poses the challenge of establishing relationships between formal variations (the syntactic dimension) and architectural meaning (semantic dimension), it also provides rich ground to discuss the thorny issue of defining design criteria. No need to remind that the question of what is an acceptable solution for a design problem is certainly debatable. After all, design problems are considered wicked precisely because the set of criteria adopted to accept or reject a design solution is unstable, open to subjective judgment and to cultural and historical contingencies (Buchannan, 1992). Hence, just like the conception of teaching exercises in design education, the conception of generative systems and the algorithms that govern them - since they operate with embedded the design criteria - must be seen as design acts in themselves.

This work has received the support of CNPq as part of the research "SUPERGRID - a kit-of-parts pedagogy. Development of a control mechanism for a computational system envisioning the design of urban façades" developed with research group LAMO as part of the Post Graduate Program in Urbanism - PROURB - of the Federal University of Rio de Janeiro.

REFERENCES

Aureli, P V 2014, 'The Dom-ino Problem: Questioning the Architecture of Domestic Space', *Log*, 30(30), pp. 153-168

Beirão, J, Duarte, J and Stouffs, R 2009, 'Grammars of designs and grammars for designing. Grammar-based patterns for urban design.', *Proceedings 23rd International eCAADe Conf.*, 1, pp. 890-904

Buchanan, R 1992, 'Wicked Problems in Design Thinking', *Design Studies*, 8(2), pp. 5-21

Carpo, M 2014, 'The Digital : From Complexity To Simplicity – And Back', *SAJ*, 6(3), pp. 256-265

Duarte, J P 2005, 'A discursive grammar for customizing mass housing: The case of Siza', *Automation in Construction*, 14(2 SPEC. ISS.), pp. 265-275

Eloy, S and Duarte, JP 2014, 'Inferring a shape grammar: Translating designer's knowledge', *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, 28(2), pp. 153-168

Engel, P 2018, 'Supergrid – A Grammar for a Kit-of-Parts Pedagogy', *4th International Symposium Formal Methods in Architecture. Unpublished.*, -, p. -

Engel, P and Eskinazi, M 2016, 'Por Uma Arquitetura Elementar', *Revista Thésis. Associação Nacional de Pesquisa em Arquitetura*, 3, pp. 54-76

Garcia, S 2017, 'Classifications of Shape Grammars.', *Design Computing and Cognition*, -, pp. 229-248

Hou, D and Stouffs, R 2018, 'An algorithmic design grammar for problem solving', *Automation in Construction*, 94(August), pp. 417-437

de Klerk, R and Beirão, J 2016 'Ontologies and Shape Grammars - A Relational Overview Towards Semantic Design Systems', *SPATIAL REASONING AND ONTOLOGIES - Volume 2 - eCAADe 34*, pp. 305-314

Kowalsky, R 1979 'Algorithm = Logic + Control', *Communications of the Association for Computing Machinery*

Love, T 2003, 'Kit-of-Parts Conceptualism: Abstracting Architecture in the American Academy', *Harvard Design Magazine*, 19, pp. 40-47

Stiny, G 1993 'Emergence and continuity in shape grammars', *Proceedings of the Fifth International Conference on Computer-Aided Architectural Design Futures. CAADFutures '93*, pp. 37-54