Title:
STG-framework: A Pattern-Based Algorithmic Framework for Developing Generative Model of Parametric Architectural Design at Conceptual Design Stage

Authors:
Chieh-Jen Lin, t60011@mail.tut.edu.tw, Tainan University of Technology

Keywords:
Design Intention, Parametric Design, Generative Modeling, Design Pattern, Algorithmic Framework

DOI: 10.14733/cadconfP.2017.288-292

Introduction:
Confusion concerning methods, thinking, and techniques among parametric, generative, and algorithmic approaches has emerged with the appearance of more new digital design tools like Grasshopper and Dynamo. Leach claims that one reason for this confusion is that the architectural domain were unfamiliar with the computer science [8], and alludes to the differences between parametric and algorithmic design, where parametric techniques are based on the manipulation of geometric forms, while algorithmic design is based on the use of programming codes. But regardless of whether through the manipulation of forms or the use of code, architects should be able to use digital architectural design tools to solve architectural design problems. Kotnik proposed that digital architectural design involves "exploring computable functions," which should take design information as input parameters and buildings' properties as output variables [7]. Literally, parametric design implies that the algorithm is fixed, and that output variables are consequently predictable from parameters. Generative modeling implies that output variables are not only controlled by input parameters, but also by flexible and adjustable functions. However, the computable functions of various architectural disciplines, in another word, the algorithms for solving various architectural design problems, should be the key to many digital architectural design issues.

In computer science domain, an algorithm is a process for solving a problem in a finite number of steps. Algorithmic modeling like Grasshopper was developed to automate and accelerate 3D modeling tasks by applying generative algorithm. But cognitive research has revealed that designers prefer to apply algorithms only as a means of exploring geometric intentions, but prefer to apply known solutions and design patterns for other non-geometric issues [14]. When designers' intentions go beyond geometry, regardless of the type of design objective [3], designers need to find or develop appropriate algorithms before they can implement generative or evaluative scripts. Algorithmic modeling is gradually applied in generating complex forms, multiple objectives optimization, as well as controlling and evaluations of buildings' performances. One of the reasons is that the relevant algorithms, including mathematical formulas of complex geometries, metaheuristic algorithms of artificial intelligent [13], structural analysis, and energy consumption formulas, have been validated in those domains. Thus the tasks of algorithmic design become to implement algorithms in modeling tools, rather than to interpret architectural design problems and derive solving algorithms. Since there are insufficient instructions and assistance for converting architectural knowledge into algorithmic scripts, it is not surprising that designers prefer to apply known solutions, rather than develop or implement algorithmic scripts on their own.

While fewer architects are directly employing the contents of Alexander's pattern language [1], more software engineers are applying "design patterns" in identifying and reusing the best practices in known situations. Based on design patterns, such as the model-view-controller (MVC) pattern, application frameworks have therefore been developed to facilitate and accelerate the development of

applications. For example, Ruby on Rails, Symfony, and Django have been developed for web-based applications in different programming languages. While programming/scripting skills have become more critical when applying algorithmic tools in parametric architectural design, developing an algorithmic framework for the exploration and development of algorithms can help architects to focus on solving design problems, rather than on programming/scripting tasks. Based on previous studies proposing information conversion patterns employing Building Information Model (BIM) schema, which involve semantic, topological, and geometric information [9], this paper proposes an algorithmic framework to help architects to explore and develop algorithms going beyond geometric intentions.

Main Ideas:
Unlike other studies focusing on algorithmic programming patterns [12], or architects' geometric intentions [10], this paper proposes an algorithmic framework that aims to help architects for connecting abstract design intentions by integrating semantic ontology and topological algorithms. Based on the STG pattern proposed in previous studies [9], this framework is divided into three parts: (1) a semantic module that can help architects to indicate design objects and their semantic relations as the "Model" module in a MVC pattern, (2) a topological module that presents the topological algorithms of given semantic relations as the "Controller" module in a MVC pattern, and (3) a geometric module that presents the visual validation of topological algorithms as the "View" module in a MVC pattern. This "Semantic-Topological-Geometric framework" (STG*f*) realizes an algorithmic framework by applying Grasshopper and the GhPython plugin as an algorithm-aided design tool [11].

The Semantic Components as Representing Models of Design Intensions
Semantic ontology is a knowledge-representation technique in the artificial intelligence (AI) domain. One of the most popular tools for authoring semantic ontology is Protégé, which is based on the Ontology Web Language (OWL) originally used to develop semantic networks. By applying OWL reasoner plugins, such as FaCT++ and HermiT, Protégé can validate an OWL-based ontology in order to ensure that it is correct and consistent. Furthermore, the semantic web rule language (SWRL) plugin can be used to express logic rules in order to infer implicit knowledge within the ontology. To convert the semantic ontology of architects' design intensions into generative algorithms, an ontological technique based on Protégé was incorporated into the semantic module of STG*f*.

The first module of STG*f* is the "Semantic" component, which consists of the semantic information of design intentions. In an MVC-based application, the Model module is used to capture behavior and logical rules in the problem domain. In order to associate generative algorithms with architectural design intentions, it must first represent design intentions in a computable format. At early design stages, design intentions usually consist of abstract, textual descriptions concerning various design objects and their relationships, and essential semantic information regarding building components can be predefined in BIM applications and Industry Foundation Classes (IFC) schema. However, although Rhino has no predefined semantic schema for building components in architectural design, an architect may define or interpret his/her unique design objects and relationships, which cannot be predicted by BIM or IFC during early design stages, when design situations have not yet emerged. Rhino therefore needs a contextual semantic ontology [5], which is a computational format for representing, storing, and validating a semantic ontology of domain knowledge in order to convert architects' design intensions into generative parametric design algorithms.

The Topological Modules as Generative Controllers of Design Intensions
Although ontological techniques can validate the conceptual consistency of design intentions, they cannot guarantee that relevant instances of semantic concepts will also comply with the necessary properties. For example, a partial ontology may indicate that an "OpenSpace" class has an "ExistingTree" property allowing the inference of the "GoodQuality" property of the "StaticLeisureActivity" class for seniors. However, it is also necessary to calculate the correlations between the existing trees and the instance of the park in order to determine whether those existing trees are located within the scope of the park. As a consequence, in addition to their definitions of conceptual classes within the ontology, topological relationships must also address more feature properties in specific instances.

The second module of STG*f* is a "Topological" component, which is a controlling algorithm for validating design intentions. In a MVC-based application, the "Controller" module is used to accept operations from users to modify the data within models, and therefore controls interactive behavior among different models in a system. Eastman suggested that topologies are the mathematical relationships and fundamental definitions of parametric models in BIM [4]. At an early design stage, topological relations expressing design intentions are usually abstract, and may consist of enclosure, extension, and concentration of indoor/outdoor spaces and building masses [6]. Topological relations among design objects defined within the "Semantic" module can thus be regarded as the "Controller" of design intentions.

The Geometric Modules as Validating Views of Design Intensions

Since Grasshopper aims to generate 3D models through given algorithms, it seems unnecessary to add other geometric functions. To ensure visual validation of whether a design intention has been accomplished, however, it should provide more visual clues for users, such as textual and numerical tags and colored previews. The final module of STG*f* is the "Geometric" component, which can validate views of design intentions. In an MVC-based application, a "View" module is used to display information concerning a retrieved "Model" and the results of "Controller." In architectural design, architects always need visual feedback to validate the content of semantic ontologies or topological behaviors of generative algorithms. By helping users to validate algorithms, immediate visual feedback concerning generative algorithms is one of the most attractive features of Grasshopper. For design intentions other than geometric features, especially in the case of invisible or non-obvious intentions like outdoors spaces or mathematical ratios, geometric features can not only be input as parameters in generative algorithms, but also be generated for the visual validation of design intentions. The "Geometric" module of STG*f* therefore aims to demonstrate how to input geometric objects from Rhino into the STG*f* "Semantic" and "Topological" modules, and how to provide visual clues for the validation of users' design intentions.

Initial Implementation of STGf

By providing rewritable example scripts and adjustable algorithmic modules, which are editable clusters of algorithmic components in Grasshopper, the STG*f* aims to help architects to explore their abstract intentions beyond geometric features at early design stages. For example, "community-friendliness" issues have been the design subjects in Taiwan's architect qualification exam for three consecutive years (Fig. 1). "Community-friendliness" issues concern how to arrange a building to form high-quality spaces for facilitating community activities. However, the outdoors spaces shaped by a building and its surroundings are usually ignored in the semantic ontology of BIM or IFC. In addition, what features of a outdoor space can facilitate community activities still leave much room for architects to interpret.

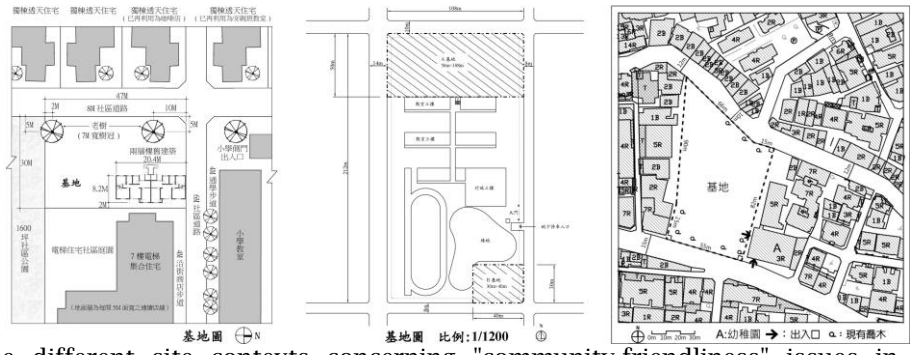

Fig. 1: Three different site contexts concerning "community-friendliness" issues in an architect qualification exam in Taiwan: (a) an architect firm as the cornerstone of a good neighborhood in 2014, (b) a community-friendly elementary school in 2015, and (c) a community library and public spaces in 2016.

As an example, a candidate may propose the idea of preserving existing trees in order to rapidly form a static leisure park for seniors, but another candidate may suggest the installation of more game and sports equipment in order to improve the health of both seniors and children. If such design intentions can be represented in OWL, then the logic reasoners in Protégé can help architects to validate the ontology. By applying the semantic components of STG*f*, architects can hook semantic ontologies from Protégé with Grasshopper's generative algorithms. By the help of Protégé, STG*f* allows abstract design intentions to be converted into a computational format, enabling them to be input as the parameters of generative algorithms.

Although ontological techniques can validate the conceptual consistency of design intentions, they cannot guarantee that all instances of semantic concepts will also comply with the necessary properties. For the idea of preserving existing trees, it is necessary to calculate the correlations between the existing trees and the instance of the park in order to determine whether those existing trees are located within the scope of the park. As a consequence, in addition to their definitions of conceptual classes within an intentional ontology, topological relationships must also address more feature properties in specific instances. By applying the topological components of STG*f*, which are rewritable example scripts and adjustable algorithmic modules, STG*f* can help architects to develop more generative algorithms for exploring topological relations of their design intentions.

To ensure visual validation of whether a design intention has been accomplished, however, Grasshopper may need more visual clues for users, such as textual and numerical tags and colored previews. The geometric components of STG*f* therefore aim to demonstrate how to input geometric objects from Rhino into the STG*f* "Semantic" and "Topological" modules, and how to provide visual clues for the validation of users' design intentions. As an example, the selection of the parking entrance is based on two design intentions: (1) traffic on the narrower street is relatively small, and (2) the entrance close the start point of the construction line can reduce pedestrian conflicts. After implementing those intentions in STG*f*, users can input the contour of the site and the borderlines of the streets around the site in order to generate a tag on the site for suggesting the location of the parking entrance into the site (Fig. 2).

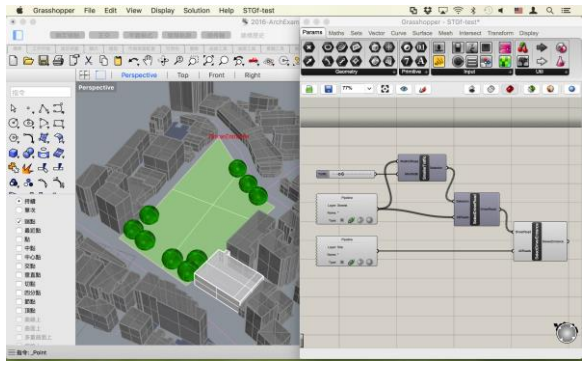

Fig. 2: An example algorithm is used to select a parking entrance into the site for reducing traffic conflicts.

One of the major obstacles to applying generative modeling is that stakeholders cannot understand the generative algorithms, especially when algorithms are too complex to be explained even by the script authors themselves. The "Semantic" and the "Topological" modules can therefore help to associate algorithms developed within generative modeling tools like Grasshopper with the architectural design intentions that were applied within those algorithms. Based on the "geometric-topological-geometric" pattern, this paper proposes an algorithmic framework termed STG*f* for modeling design intentions that go beyond geometric forms.

Conclusions:

While critics have suggested that the use of generative tools can result in the over complexity of simple things [2], generative algorithms should potentially be able to go beyond geometric intentions.

However, the programming and information technology knowledge usually becomes the biggest obstacle for architects who wish to implement algorithms of their intentions beyond geometric knowledge. As software frameworks can dramatically simplify and accelerate the development of an application, the STG*f* framework proposed in this paper can also help designers to simplify and accelerate the development of generative algorithms used in parametric architectural design.

One purpose of the MVC pattern is to divide programming tasks of a complex system into independent objects. The STG*f* framework divides parametric design into three algorithmic procedural steps, and can implement generative algorithms by different designers/scripters. As building projects become more complex, instead of requiring architects to wear many hats associated with other domains, it will be better to hand over programming/scripting tasks to professional scripters, and performance optimization to MEP engineers. It is therefore time to embed architects' design intentions in the parameters, variables, and algorithms used in parametric architectural design.

Acknowledgements:
The Ministry of Science and Technology of Taiwan support this paper under grant number MOST 105-2221-E-165-002.

References:
[1] Alexander, C.; Ishikawa, S.; Silverstein, M.: A Pattern Language : Towns, Buildings, Construction, Oxford University Press, New York, 1977.
[2] Burry, M.: Scripting Cultures: Architectural Design and Programming, John Wiley and Sons, Ltd., Chichester, UK, 2011.
[3] Chang, M.-C.; Shih, S.-G.: A Hybrid Approach of Dynamic Programming and Genetic Algorithm for Multi-criteria Optimization on Sustainable Architecture Design, Computer-Aided Design and Applications, 12(3), 2014, 310-319. http://dx.doi.org/10.1080/16864360.2014.981460
[4] Eastman, C.; Teicholz, P.; Sacks, R.; Liston, K.: BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors, 2nd ed., John Wiley & Sons Inc., Hoboken, N.J., 2011. http://dx.doi.org/10.1002/9780470261309
[5] Gursel, I.; Sariyildiz, S.; Stouffs, R.; Akin, Ö.: Contextual Ontology Support as External Knowledge Representation for Building Information Modelling, in: T. Tidafi, T. Dorta (Eds.) Joining Languages, Cultures and Visions: CAADFutures 2009, PUM, 2009, 487- 500.
[6] Ho, H.-Y.; Wang, M.-H.: Meta Form as a Parametric Design Language, in: eCAADe 2009, Istanbul, Turkey, 2009, 713-718.
[7] Kotnik, T.: Digital Architectural Design as Exploration of Computable Functions, International Journal of Architectural Computing, 8(1), 2010, 1-16. http://dx.doi.org/10.1260/1478-0771.8.1.1
[8] Leach, N.: Parametrics Explained, Next Generation Building, 1(1), 2014, 33–42.
[9] Lin, C.-J.: The STG Pattern: Application of a "Semantic-Topological-Geometric" Information Conversion Pattern to Knowledge Modeling in Architectural Conceptual Design, in: Proceedings of the 21st International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA 2016), Melbourne, 2016, 435-444.
[10] Su, H.-P.; Chien, S.-F.: Revealing Patterns: Using parametric design patterns in building façade design workflow, in: Proceedings of the 21st International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA 2016), Melbourne, 2016, 167-176.
[11] Tedeschi, A.; Wirz, F.; Andreani, S.: AAD_Algorithms-Aided Design : parametric strategies using Grasshopper, Le Penseur Publisher, Brienza, Italy, 2014.
[12] Woodbury, R.: Elements of Parametric Design, Routledge, New York, 2010.
[13] Wortmann, T.; Nannicini, G.: Black-Box Optimisation Methods for Architectural Design, in: 21st International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA 2016), Melbourne, 2016, 177-186.
[14] Yu, R.; Gero, J.; Gu, N.: Architects' Cognitive Behaviour in Parametric Design, International Journal of Architectural Computing, 13(1), 2015, 83-102.