COMPUTER-AIDED DESIGN & APPLICATIONS, 2017
VOL. 14, NO. 3, 313–323
http://dx.doi.org/10.1080/16864360.2016.1240452

The STG pattern – application of a "Semantic-Topological-Geometric" information conversion pattern to knowledge-based modeling in architectural conceptual design

Chieh-Jen Lin

Tainan University of Technology, Taiwan

ABSTRACT

Generative modeling tools have become a popular means of composing algorithms to generate complex building forms at the conceptual design stage. However, composing algorithms in order to meet the requirements of general design criteria, and communicating those criteria with other disciplines by means of generative algorithms still faces many technical challenges. This paper proposes the use of a "Semantic-Topological-Geometric (STG)" pattern to guide architects in composing algorithms for representing, modeling, and validating design knowledge and criteria. The STG pattern can help architects to convert semantic information concerning the conditions and requirements of a project into design criteria, which are usually composed of topological relations among design elements, in order to explore the geometric properties of spaces and building components by means of generated 3D models.

KEYWORDS

Generative modeling; design criteria; design pattern; semantic ontology; BIM

1. Introduction

Generative modeling tools like Grasshopper have become a popular means of composing algorithms for generating complex building forms, optimizing multiple design objectives, and structural and sustainability control at the conceptual design stage [18]. The visual programming interfaces of Grasshopper are easier to learn and understand than textual programming tools. With the help of immediate feedback of visualized 3D models in Rhino, generative modeling tools allow architects to freely explore creative ideas expressing geometric intentions. Except in the case of constructability issues involving complex geometric forms, however, one of the major issues affecting application of generative modeling in architectural design is how to associate generative algorithms with known design criteria in order to evaluate whether the generated forms are acceptable or not. How to compose algorithms in order to meet the requirements of general design criteria, and how to communicate those criteria with other disciplines by means of generative algorithms still faces many technical challenges.

The "Pattern Language" proposed by Alexander yields conclusions concerning the good practices of endemic buildings that serve as design paradigms for acquiring the knowledge needed to solve common problems [1]. However, while few architects actually employ Alexander's language, a relatively large number of software engineers apply "design patterns" in identifying and reusing the best practices in known situations. In the software engineering domain, "design patterns" do not determine the final design of software, but instead specify methodologies for solving commonly occurring problems within a given context. Some design patterns have been accepted as best practices, such as the model-view-controller (MVC) pattern for implementing user interfaces based on object-oriented programming. With generative modeling and parametric design becoming more popular and important in architectural design, how to translate design criteria into computational procedures has become a common problem when employing generative modeling tools. However, there is still no pattern to guide architects in composing generative algorithms that can implement their design knowledge and criteria.

2. Design patterns in computational digital architecture

With the widespread penetration of digital tools into almost all areas of architectural design, including both education and practice, digital tool application skills

CONTACT Chieh-Jen Lin t60011@mail.tut.edu.tw

and knowledge have become more important than in the past. To apply generative modeling tools such as Grasshopper, a designer needs more knowledge of mathematical formulas than basic architecture knowledge [16], and more data processing skills than aesthetic skills involving geometric forms. However, the need for additional skill and knowledge often causes the results of parametric design to be disconnected from architectural contexts, such as material, user, and usage requirements, and causes designers to expend more effort on programming/scripting of algorithms than on architectural design [18].

Grasshopper is a graphic algorithm editor integrated with Rhino that can be used to explore novel geometric shapes, and cognitive studies have revealed that parametric design mainly supports designers' geometric intentions [18]. Beyond geometric intentions, designers also tend to select existing solutions for known problems, instead of developing new solutions, which meets the definitions of Alexander's pattern [17]. Obviously, generative algorithms should potentially be able to go beyond geometric intentions [13]. For example, the use of generative algorithms in optimization of spatial planning [2] and building performance during early design stages has been explored [3]. Although those attempts have chiefly consisted of implementations of existing algorithms for specific design issues, such as spatial syntax for space planning, and genetic algorithms for optimizing multiple objectives concerning building performance, the results have demonstrated potential application to more general design criteria than just the realization of novel geometric intentions.

While generative modeling and parametric design has impacted architects' thinking and methodology during the early and developing design stages, BIM applications have been used to extensively improve workflow during the later and detailed design stages. Since they can manipulate more semantic and topological information concerning building components than 2D CAD or 3D models, BIM applications provide a convenient platform for visually communicating with different disciplines, especially concerning the mechanical, electrical, and plumbing (MEP) engineering of a project. Based on the design information schema of BIM, which consist of semantic, topological, and geometric information, and referring to the MVC pattern in software engineering, therefore this paper consequently proposes an algorithmic design pattern based on a "semantic-topological-geometric (STG)" framework [7], and this pattern can enable designers compose algorithms for modeling general design criteria at the early design stages.

2.1. Semantic ontology as the information model of design criteria

The first component of the STG pattern is "Semantic Ontology," which consists of the information models of design criteria. In the MVC pattern, a "model" module is the central component used to capture behavioral and logical rules in a problem domain. To associate algorithmic generative modeling with design criteria, the system must first represent design criteria in a computable format. At early design stages, architectural design criteria usually consist of abstract, textual descriptions of various requirements. In order to represent semantic criteria of architectural design into a computable format, semantic ontology was incorporated into the STG pattern.

Protégé is one of most popular tools for encoding domain knowledge in knowledge engineering [12]. Knowledge representation in Protégé is based on the Ontology Web Language (OWL), which is an XML-based language, and was originally used to develop semantic networks. Based on the semantic web rule language (SWRL), which can used to express rules as well as logic, the logic reasoner in Protégé can validate and infer implicit knowledge within an ontology. If architectural design criteria can be expressed in OWL, then a SWRL reasoner can help architects and stakeholders of a building project to validate criteria, infer implicit criteria, and keep those criteria consistent throughout the design process. For example, because an enclosed and quiet space may make children nervous, the Japanese architect Takaharu Tezuka, who had designed some award-winning kindergartens in Japan, asserted that the classrooms in a good kindergarten should have enough openness to make children feel relaxed [14]. This design criterion may be presented as two simple rules in SWRL format as follows:

$$\text{Kindergarten}(?k) \wedge \text{Classroom}(?c) \wedge \text{hasClassroom}(?k, ?c) \wedge \text{hasSpatialQuality}(?c, \text{FeelRelaxed}) \rightarrow \text{hasSpatialQuality}(?k, \text{GoodQuality}) \quad (1)$$

$$\text{Classroom}(?c) \wedge \text{hasOpenness}(?c, \text{HighlyOpen}) \rightarrow \text{hasSpatialQuality}(?c, \text{FeelRelaxed}) \quad (2)$$

The Criterion 2 means that an instance of "Classroom" class, which has a "HighlyOpen" instance of "Openness" property, implies this "Classroom" instance has a "Feel-Relaxed" instance of "SpatialQuality" class. However, many textural criteria will not indicate how to implement or validate the criterion, and this criterion also doesn't

indicate how to implement or validate the "HighlyOpen" property of a "Classroom." Tezuka therefore designed the "Fuji Kindergarten," an award-winning kindergarten in Tokyo, which has four "teaching spaces" as classrooms enclosed only by sliding patio doors but without walls (Fig. 1). In this context, the implementation of the "Openness" criterion can be presented as a simple rule of SWRL format as follows (Criterion 3):

$$\text{Classroom}(?c) \wedge \text{beEnclosedBy}(?c,\ \text{PatioDoors}) \rightarrow \text{hasOpenness}(?c,\ \text{HighlyOpen}) \quad (3)$$

The Criterion 3 means that an instance of the "Classroom" class, which is encloded by the "PatioDoors" instance of the "Opening" class, implies this "Classroom" instance has a "HighlyOpen" instance of "Openness" property. Basically, the "beEnclosedBy" class is an "Object Property," which is used to declare semantic relations between different conceptual classes in Protégé. In an OWL triple, "Object Property" is a "predicate" for connecting the subject and the object of a knowledge chunk within an ontology. Once a predicate has been used to connect two architectural design objects, such as "Classroom," "Opening," or other kind of architectural design components, the predicate can be used to indicate a topological function waiting for implementing in the next step of the STG pattern. As the "Model" module in the MVC pattern, a semantic ontology of design criteria can therefore not only play the role of information models for storing and representing more general design criteria proposed by architects, but also provide a guide for scripters when composing algorithms for generating proposals and validating general design criteria rather than geometric intentions.

2.2. Topological relations as controlling algorithms of design criteria

The second component of the STG pattern is "Topological Relation," which is a controlling algorithm for design criteria. Eastman declared that topologies are the fundamental definitions of parametric models in BIM [4]. At an early design stage, the topological relations of design criteria are usually abstract, and may consist of enclosure, extension, and concentration of indoor/outdoor spaces and building masses [5]. Except for some geometric rules, such as the constraints in Revit, abstract topologies for early design stages are absent from most BIM applications, however.

In the MVC pattern, a "Controller" module is used to accept operations from users to modify the data within models. A "Controller" can therefore control the interactive behavior among different "models" in a system. Topological relations among design criteria can thus be regarded as the "Controller" of design criteria. In the case of the "Fuji Kindergarten" by Tezuka, for example, the "SpatialQuality" property of a "Classroom" instance is implied by the "Openness" property, and the "Openness" property is implied by the "beEnclosedBy" property of a "Classroom" instance. Accordingly, the "beEnclosedBy" function, or "Encloses" function, which is the inverse property of "beEnclosedBy," becomes a topological function waiting for implementation in order to validate the "Openness" property. Through the help of SPARQL, which is a "SQL like Protocol and RDF Query Language" integrated in Protégé for the retrieval and manipulation of RDF data, users can implement information queries by calculating data. However, it is difficult to manipulate the geometric features of 3D models by a data query language like SPARQL. Grasshopper is therefore a better and easier

(a) (b)

Figure 1. The classrooms of the Fuji Kindergarten designed by Takaharu Tezuka: (a) Classrooms are enclosed by patio doors but without walls, and (b) The patio doors can be fully opened to remove the boundaries of classrooms [9].

tool for implementing topological interactions between semantic information and the geometric features of 3D model.

As an example, when implementing the "Encloses" topology, a simple algorithm can be proposed as follows: (1) draw at least two separate and unconnected curves as walls of a space, (2) connect the closest endpoints of each walls as the openings of the space, and (3) the generating shape enclosed by walls and openings is the space enclosed by the drawn walls (Fig. 2). This algorithm can easily be implemented by the "Connect Curves" component of Grasshopper to generate a spatial shape by inputting some curves. Although this simple "Encloses" algorithm needs the geometric features of walls as input parameters, in the case of most topological relations, however, different shapes and numbers of walls may derive out similar "Encloses" topology. The controlling behaviors of topological functions in a STG pattern are therefore more dependent on the composed algorithms than on the input geometric features of models.

Furthermore, the "Openness" property of a space can be easily generated from the "EnclosedRate" property, which is the inverse property of "Openness" is derived by dividing the total length of walls by the perimeter of the space. For example, a designer can define a rule for declaring the "Openness" instance: (1) A "Highly-Open" instance when "EnclosedRate" is less than 0.4, (2) an "Enclosed" instance when "EnclosedRate" is more than 0.6, and (3) a "SemiOpen" instance when "EnclosedRate" is between 0.4 and 0.6. However, these algorithms of the "EnclosedRate" property can ignore the geometric features of space by simplifying the length of a wall as the straight-line distance of two endpoints, as in the case of the length of an opening. The three differently-shaped spaces in Fig. 3 can therefore be derived out same "SemiOpen" instance of "Openness" property (Fig. 3). As architects usually ignore the details of geometric features at early design stages, it is better to minimize the interference of geometric features when developing algorithms for manipulating topological relationship in order to keep enough geometric manipulations freedom for the exploration of possible proposals. By retrieving predicates from the semantic ontology of design criteria, the STG pattern can guide Grasshopper scripters in determining which criterion has what topological functions.

2.3. Geometric features as validating views of design criteria

The final component of the STG pattern is the "Geometric Feature," which can visually represent and validate design criteria. In a MVC pattern, a "View" module is used to visualize a retrieved "Model" and the results of "Controller." In architectural design, architects always need visual feedbacks to validate the instances of design proposals, and to determine the behaviors of topological algorithms. Because immediate and visual feedback is one of the most attractive features of Grasshopper for architects, it is not surprising that most designers chiefly apply Grasshopper in the exploration of geometric intentions. It is more important for architects to validate other intentions, such as semantic and topological criteria, rather than geometric intentions. Although the geometric features of proposed models are usually the

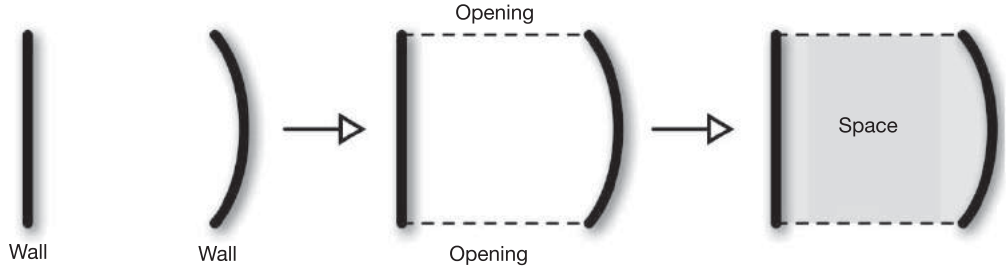


Figure 2. A sample algorithm for implementing an "Encloses" topology by using geometric features: (a) First draw two or more curves as walls, (b) then connect the closest endpoints of walls as openings, and (c) finally generate an enclosed shape using those walls and opening.

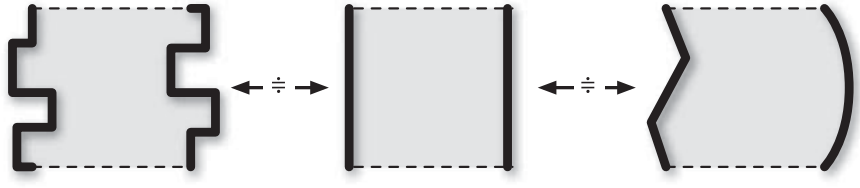


Figure 3. Three different shapes of spaces have the same "SemiOpen" instance of "Openness" property derived from a simplified algorithm, which calculates "EnclosedRate" while ignoring the shape of walls.

(a) (b)

Figure 4. Geometric features of the roof of the Fuji Kindergarten designed by Takaharu Tezuka: (a) The geometric features of the roof consist of two simple ellipses, and (b) a series of mock-up models created by the architect [9].

critical parameters for validating design criteria, those features usually cannot be easily generated from semantic or topological criteria. Geometric forms are ultimately within the domain of an architect's creativity in representing design concepts.

In the case of the "Fuji Kindergarten" by Tezuka, the initial concept of the building was an annular shape around a playground (Fig. 4a), which occurred to the architect when he recalled that his children liked to run around a circle. However, a series of mock-up models created by the architect revealed some situations, such as existing buildings and trees within the site (Fig. 4b), which needed to be considered when the architect explored geometric features. Some peculiar shapes emerged when trying to fit the building into the irregular shape of the available land, while avoiding existing buildings and trees based on general design criteria. To realize implementing the idea of an annular shape, the architect finally chose to allow the existing trees to grow directly through the roof in order to keep the building a perfect eclipse. Except for the previously-established design criteria, the situations within/around the site are the major constraints on geometric intentions. The geometric features of the site situations, such as the site's shape, terrain, vegetation, and existing buildings, therefore became the geometric features for modeling their relevant criteria. For example, a semantic criterion can be declared so that an instance of "ExistingTree" implies an instance of "OutdoorSpace" around the tree, thus the "hasMinimumSpace" topology will be ready for implementation. The geometric features of design objects, include the site situations and the architect's proposals, can not only serve as the validating views of established design criteria, but also the visual hints for modeling more semantic criteria.

2.4. Summary

One of the major obstacles to applying algorithmic generative modeling is that stakeholders cannot understand the algorithms, especially in the case of those algorithms too complex to be explained by the designers who are actually implementing those algorithms. The "Model" of semantic ontology and the "Controller" of topological relations of design criteria can therefore help in associating design proposals, which are generated by algorithmic modeling tools like Grasshopper, with the architectural design knowledge that was applied within those algorithms. This approach does not attempt to implement rapid modeling of design concepts, but rather aims to divide scripting tasks into small and independent parts, which can be assigned to different participants, who may hold different skills, knowledge, and responsibilities. For modeling more general design criteria than geometric intentions, it is necessary to translate different types and sources of design criteria into computable formats like semantic ontology before composing the generating and validating algorithms. The three steps of the STG pattern can therefore not only guide task assignment procedures, but also facilitate communication with relevant stakeholders by means of easily-readable formats of textual criteria formats within the semantic ontology.

3. Implementation and initial testing of the STG pattern

Unlike Alexander's pattern language, which can package instances of solutions with relevant design problems, a programming design pattern should not only provide direct solutions to known problems, but also meta-knowledge for identifying design situations, and then

select appropriate methods for addressing those problems. In the wake of a "geometric-topological-geometric" information conversion framework [7], this paper proposes an algorithmic pattern for modeling general design criteria that goes beyond the geometric intentions of a building's form. By taking the Fuji Kindergarten by the architect Tezuka as an example, the STG pattern demonstrates how to guide designers in associating generative modeling with their architectural knowledge when composing algorithms.

3.1. Reverse modeling semantic ontology

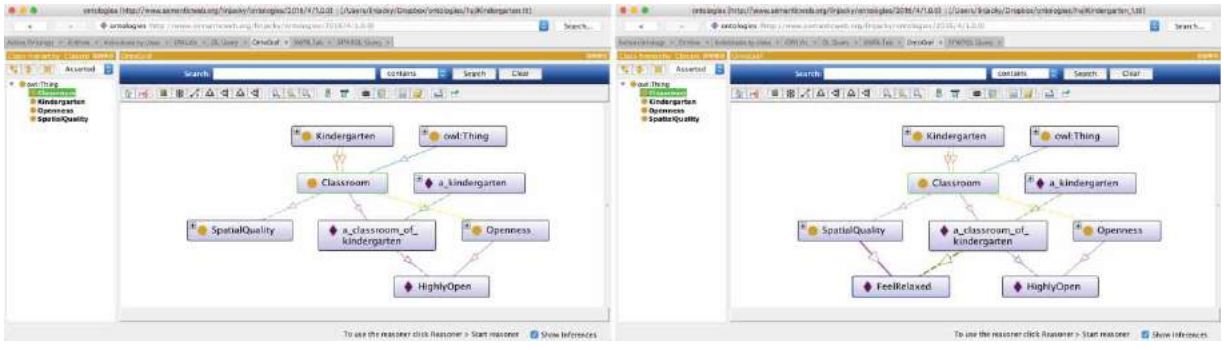
Since the graphic scripting interface of Grasshopper provides a convenient tool for manipulating the geometric features of 3D models, and composing algorithms for validating topological relations among the features of those models, the first critical tasks in implementing STG pattern is to integrate semantic ontology techniques into Grasshopper. In previous studies, a prototype Python scripts, which was entitled "Design Criteria Modeling (DCM)," which sought to help designers to model and validate semantic ontology within Grasshopper by hooking with OWL files and the SWRL reasoner of Protégé, was implemented [7]. Because of an incompatibility issue of Python-based OWL reasoner, most manipulations of semantic ontologies in DCM still elies on the Protégé, and was usually too complex to be recognized and learned by designers.

For example, before a designer can declare simple design criteria like Rule 1 to 3, he/she must model the ontology in Protégé as follows: (1) Declare the conceptual classes of all of design objects, such as "Kindergarten," "Classroom," and "SpatialQuility"; (2) declare the object properties of those classes, such as "hasClassroom," "hasSpatialQuality," and "hasOpenness;" (3) declare the object properties between relevant classes, such as "Kindergarten" "hasClassroom" some "Classroom", and (4) finally declare the individuals/instances of relevant classes and properties in order to write the reasoning rules of relevant individuals/instances (Fig. 5). Based on DCM, this paper therefore implements a Python-based script for helping designers to easily present design criterion rules, and to reversely generate the relevant semantic classes and object properties for constructing an ontology of those design criteria.

A SWRL sentence in human readable syntax consists two basic types of clauses: (1) Classe(?c) clause, such as "Classroom(?c)," which means an instance varible "?c" of "Classroom" class, and (2) Property(?c, instance) clause, such as "hasOpenness(?c, HighlyOpen)," which means an instance "?c" has an instance of "hasOpenness" property which is the "HighlyOpen" instance. Therefore, once a rule sentence has the rule 2 input in to the ontology module of STG, it will reversely generate a "Classroom" object, and the "hasOpenness" and "hasSpatialQuality" properties, and further acquires the actual class names of instances "HighlyOpen" and "FeelRelaxed" from the user. By recognizing the second parameter of a clause, STG implements the reverse modeling of an ontology of design criteria.

3.2. Example topological algorithms

After the ontology of design criteria has been constructed, the next step in the STG process is to compose topological algorithms for generating or validating the topological relations of design criteria, which are usually the object properties for connecting design objects in the ontology, such as the "BeEnclosedBy" or "Enclosed" property and the "Opening" class of the "HighlyOpen" instance. As mentioned above, the algorithm of ""BeEnclosedBy" and "Openness" can be implemented using the "Connect Curves" component of Grasshopper in order to generate a spatial shape. However, the manipulation of input and generated geometric data often poses technical difficulties.



(a) (b)

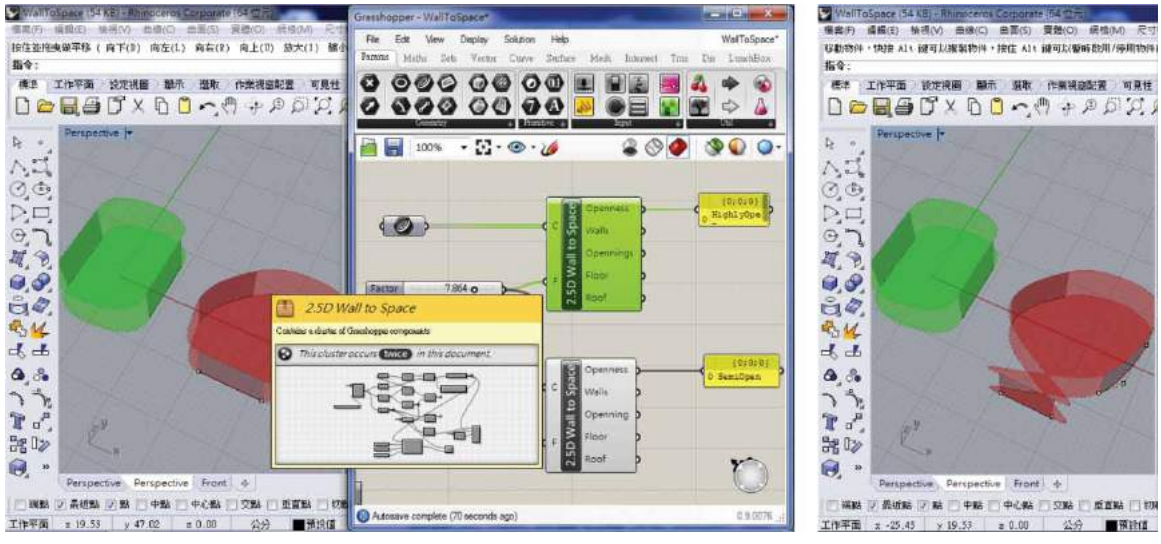
Figure 5. (a) Visualization of relevant classes in the ontology of Rule 1 and 2 design criteria, and (b) the "Relaxed" instances of "SpatialQuality" inferred by the SWRL reasoner in Protégé.

For example, the "Connect Curves" component of Grasshopper can easily generate an enclosed shape in order to represent a space by inputting several separate and unconnected curves (Fig. 6a). However, the drawing directions and the order of selection of input curves directly affect the generating shape, which sometimes violates the designers' expectations (Fig. 6b). Another difficulty is how to obtain the length of the generated openings, which should be the total length of the new generated segments other than the input segments retrieved from the enclosed shape. One solution is to apply the "Explode Curve" component, and then to retrieve the even items from the list of exploded segments using the "Cull Pattern" component. Since designers are often not familiar with data structures and programming skills, which are the critic foundation of generative design, the implementation of topological algorithms usually requires extensive guidance from experienced Grasshopper instructors, or the examination of similar algorithms from existing examples. Not like Lunchbox, which packages complex algorithms into un-modifiable components, the STG pattern proposes some example component clusters, which comprise a set of well pre-assembled components for demonstrating basic topologies such as "Connects," "Encloses," and "Avoids." A "Cluster" is the means by which Grasshopper simplifies the display of complex combinations, which can be repeatedly reused as new algorithmic components many times in a GH file, and can protect the combination by passwords from unauthorized changes. Since those clusters aren't protected by passwords, so it is easy for designers to learn, copy, or modify the algorithms within them in order to develop new combinations for resolving new topologies.

3.3. Filtering geometries for design criteria

Architectural design usually ignores some geometric details at the early stages. Conversely, the input geometries in at the early stage are usually applied to represent or develop important design criteria. For example, the contour lines of a building site imply the maximum range of the available land, and the input situations with/around the site, such as adjoining roads, existing trees and buildings, imply different implicit design criteria, such the "Avoids" property of trees, and the "Retreats" or "OutdoosSpaceAround" property of the buildings, which must await for modeling the semantic ontology or implementation of topological functions. Since those geometries were input by the user, however, the identification of semantic relations between geometries and design criteria usually imposes a burden on the user.

One advantage of a BIM application is that most geometries have their default building component semantics, which can indicate which geometries are the instances of what components, such as the "Wall," "Floor," "Roof," or "Room" classes. Except for some plugins, such as visualARQ [15], which is a commercial plugin for architectural design, however, all geometries in Rhino have no default semantic features connecting architectural design objects. The STG pattern therefore applies the "Pipeline" component of Grasshopper, which is a data filter that can automatically collect geometries from Rhino by types, layers, and names of geometries.



(a) (b)

Figure 6. (a) A "2DWallToSpace" cluster implements the "Openness" topology to identify a "HighlyOpen" instance and "SemiOpen" instance of two spaces, but (b) the directions and the order of the selected curves directly affect the generating shape of a space.

The STG pattern therefore has some layers with default names, such as "Site," "Floor," "Wall," and "Roof," as semantic clues for the input geometries of relevant design criteria.

3.4. Initial test of the STG pattern

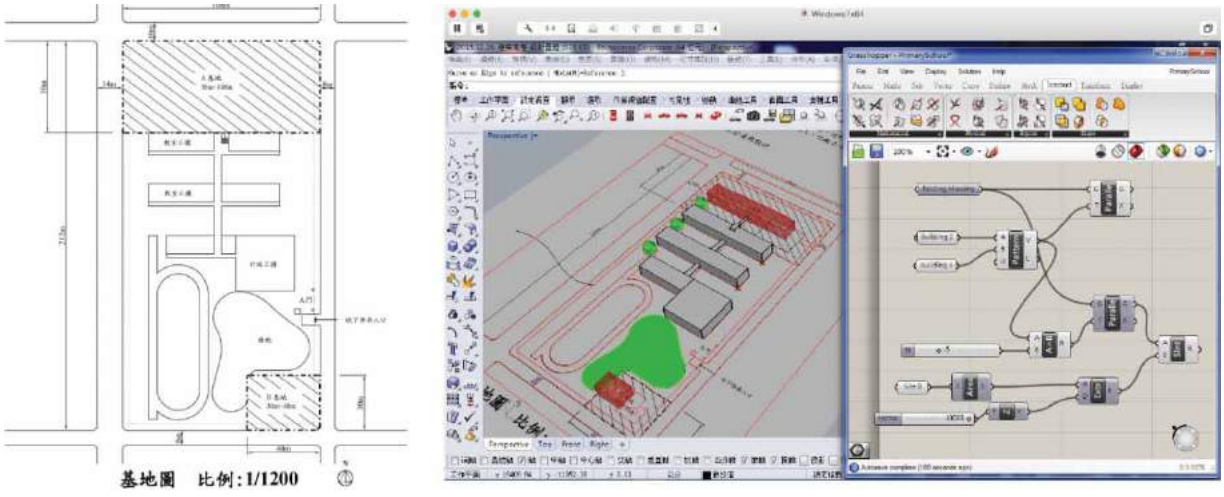
The DCM prototype was provided to students for modeling their semantic design criteria. They were asked to rapidly design a "Community-Friendly Primary School," which was a topic on Taiwan's architect qualification exam in 2015. The design context consisted of two sites located on north and south sides of a primary school (Figure 7.a), and the building's purpose was for the learning and leisure use of seniors in the community. Except for such basic issues as the building code, traffic, and climate response, the existing site context, including existing trees, classrooms, exercise yard, and green areas, served as the predominant element for the development of design criteria.

In this test, most of the students found the strong patterns of existing buildings, including an axis formed by the central corridor, and the rhythm formed by parallel building masses. They therefore tended to follow the axis by extending the central corridor to connect both sites, and then arranged the new building to be parallel with existing buildings. The students could consequently first code the "Parallel" and "Axis" topology, then seek to implement the algorithms of "Parallel" and "Axis," and finally select the input parameters, such as an existing building and its gaps as the generating rules (Figure 7.b).

4. Discussion

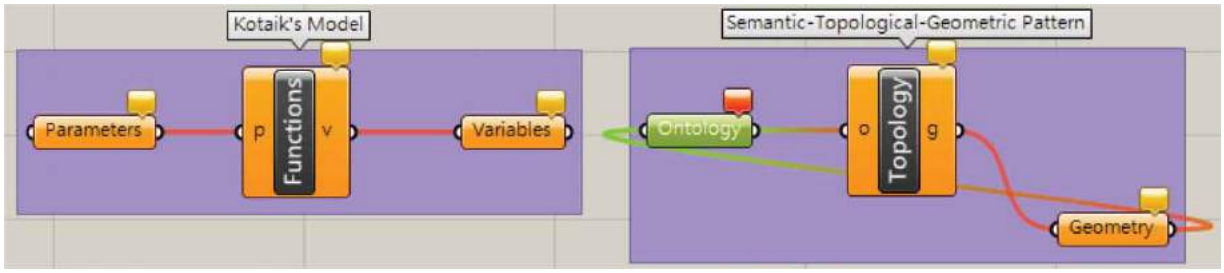
New design thinking and methods have emerged as more digital tools penetrate architectural design education and practices. However, because digital tools require considerable prior knowledge apart from basic architectural knowledge, many misunderstandings concerning digital architectural design have also arrived with the increased availability of digital tools [11]. Confusion about methods, thinking, and modeling among generative, parametric, and algorithmic approaches has emerged with more new digital tools. Kotaik therefore proposes a theoretical framework that addresses the differences among the generative, parametric, and algorithmic approaches in terms of computability [6].

Kotaik concludes that a critical feature of digital architectural design is the ability to explore the computable functions of various architectural disciplines. Kotaik remapped Oxman's five class models of digital design, which are CAD, formation, performance, generative, and integrated compound models [10], onto three levels of design computability, which are representational, parametric, and algorithmic in the order of low to high computability. Furthermore, a formal description of digital architectural design has been proposed as a computable function for generating "variables" of architectural properties by inputting "parameters" of the architectural factors of buildings or their environment. Kotaik's model of digital architectural design therefore be translated into a Grasshopper-style visual format as shown in Fig. 8a, and the STG pattern can be represented as Fig. 8b. The obvious difference between Kotaik's model and STG pattern



(a) (b)

Figure 7. The site contexts for the rapidly design of a "Community-Friendly Primary School": (a) The context and content of the site, and (b) a test model for developing design criteria.



(a) (b)

Figure 8. Two conceptual models of computing architectural design: (a) Kotaik's model (Left), and (b) the STG pattern (Right).

is that the generated geometric forms may recursively generate or be input as new semantic ontologies of design criteria. The STG pattern is discussed as follows based on this computability model of digital architectural design (see Fig. 8).

4.1. Semantic ontology as the input parameters of computable architectural design

Woodbury defined parametric design thinking as thinking with abstraction, mathematically, and algorithmically [16]. In the case of architectural design, algorithmic thinking means to understand the interpretive correlations between algorithms and design criteria [2]. However, most abstract design criteria at early design stages, such as the "Openness" property of classrooms in the Fuji Kindergarten, are too abstract to be directly input as parameters, and it is not easy to find an algorithm for generating or validating them. For instance, designers cannot input a "FeelRelaxed" instance of "SpatialQuality" property into a function for generating the "HighlyOpen" instance of a classroom.

Kotnik indicated the output "Variables" should be an architectural property, and might include a building's geometric form or components. However, the possible input "Parameters" of computable functions are left with open definitions. If the input parameters of a computable function cannot be associated with any known architectural design criteria, in view of the principle of "garbage-in, garbage-out," it is not surprising that parametric design is often criticized as being disconnected from basic architectural design knowledge.

In order to convert semantic design criteria into computable parameters to facilitate machine processing, it is necessary to represent them in a formal and computable format. With help of STG's scripts based on SWRL, it is easier to model semantic rules of design criteria than using the DCM approach, which relies on hooking Protégé by OWL files in order to employ the OWL reasoner in Protégé. Semantic criteria at the early design stages can therefore be regarded as input parameters for the algorithms of generative modeling tools like Grasshopper. However, since STG or DCM do not implement the full reasoning functions of OWL, STG still requires the help of Protégé for validating the completeness and consistency of a complex design criteria ontology.

4.2. Topological relations as controlling algorithms of computable architectural design

Generative algorithms are the core of a computable architectural design. However, a parametric model usually implies that the algorithms are fixed, and the correlations between input parameters and output variables are consequently predictable. Cognitive studies have found that, when solving design problems, designers put more effort into finding appropriate collections of generative algorithms fitting their intentions [18], than testing possible output variables by modifying the input parameters. The extra costs of applying parametric modeling usually do not mean those design problems are so novel that the solution algorithms have not been invented, but usually only mean that the designer does not know the solution algorithms or has not mastered those algorithms.

Visual programming in Grasshopper should not requires excessive programming knowledge and skill. However, when designers' intentions become more complex, programming knowledge, including mathematical knowledge, data structure analysis and programming skills, becomes critical in developing acceptable algorithms. Consequently, researchers claim that parametric design imposes new challenges by asking architects to play the roles of programmers/scripters [18]. However, algorithms for modeling complex geometries have already been developed by mathematicians, and some have even been packaged as plugins of Grasshopper, such as the "LunchBox" plugin [8], which can generate a parametric Mobius strip, Klein bottle, and different shapes

subdividing given surfaces. However, it must be possible to classify those geometric intentions into aesthetic or creative criteria, rather than general architectural design criteria, which can be validated by algorithms.

The population of using the "LunchBox" plugin by architects reveals that extra effort is usually spent on converting mathematical knowledge into programming scripts in parametric design, rather than on converting architectural knowledge into generative algorithms. In the case of the Fuji Kindergarten, for example, the architect can only identify topological relations, such as connective and surrounding as the controller, and then identify classrooms and the playground as the input parameters. Thus, algorithmic coding tasks can be handed over to professional programmers/scripters. Once topological algorithms are regarded as input parameters for modifying parametric modeling, an architect's true creativity can be released from the restricted output variables of fixed algorithms.

4.3. Geometric features as visual validations of computable architectural design

The original purpose of Grasshopper was to automate and accelerate 3D modeling tasks in Rhino. However, the computability of generative algorithms should not be restricted to only the generation of complex geometries. According to Kotnik's view, a "threshold" between computable and non-computable design should be located between representational and parametric design models, and the representational applications of digital tools are only an alternative to paper drawings and hand-made models, such as those produced using conventional 2D CAD or 3D modeling software.

From the view of computability, the application of plugins like "LunchBox" to generate complex geometries is actually closer to the representational level than the parametric level. Key knowledge of architectural design is thus not contained in the generative algorithms of geometric forms, but rather in the selection of input parameters and how to filter out generated geometric forms as output variables. In the case of the Fuji Kindergarten, before the architect took an ellipse as the predominant geometry, he had to learn or develop some criteria concerning the geometric features of the playground and the building, such as the minimal required spaces of the playground and a classroom, and the necessary retreats along the contour of the existing trees and the site. Even though some of these criteria cannot be used to directly generate geometric forms, the generated geometries still can be used to visually validate those criteria. For example, a designer can apply a "hasOutdoosSpaceAround" topology to connect the existing trees, buildings, and the shape of the building site in order to generate a buildable shape within a site. By applying the "Subtract" component in the topological algorithms of "hasOutdoosSpaceAround," the geometric features of existing trees and buildings within a site can be the visual representations of design criteria. Consequently, the geometric features of generative building forms cannot only represent designers' geometric intentions, but can also visually validate the invisible design criteria of architectural design.

5. Conclusion

At present, parametric design mainly is chiefly applied to complex building form generation, multiple design solution optimization, and structural and sustainability control [17]. Parametric design is not only a novel tool for digital architectural design, but also a new methodology for architectural design thinking. Following the generative approach, Kotnik concluded that the exploration of computing functions is a critical feature of digital architectural design. However, there may be insufficient clues for discovering the computable functions of general architectural design criteria, especially for those abstract concepts proposed by architects and emerging in the early design stages. To expand the use of parametric design, this paper proposes an "STG" pattern based on a "semantic-topological-geometric" information-converting framework to guide designers in modeling design criteria knowledge in the algorithmic modeling tool Grasshopper.

In view of the fact that one purpose of the MVC pattern is to divide programming tasks in complex systems into small, discrete, and independent objects, the STG pattern divides computational architectural design into three parts characterized by computable functions, which can implement generative algorithms by different designers. As building projects become more complex, instead of requiring architects wear many hats associated with other disciplines, it is better to hand over programming/scripting tasks involving complex geometric generation and performance optimization to software and MEP engineers. It is therefore time to embed basic and conventional design knowledge into the parameters and variables of generative architectural design. Using guidance from STG, it is possible for architects to communicate design criteria with parametric modeling scripters and other disciplines of an AEC projects.

Acknowledgements

The Ministry of Science and Technology of Taiwan has supported this paper under grant number MOST 103-2221-E-165-001-MY2.

ORCID

Chieh-Jen Lin http://orcid.org/0000-0001-9981-5864

References

[1] Alexander, C.; Ishikawa, S.; Silverstein, M.: A Pattern Language : Towns, Buildings, Construction, Oxford University Press, New York, 1977.

[2] Bazalo, F.; Moleta, T.J.: Responsive Algorithms - An investigation of computational processes in early stage design, in: Proceedings of the 20th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA 2015), Daegu, 2015, 209–218.

[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] Ho, H.-Y.; Wang, M.-H.: Meta Form as a Parametric Design Language, in: eCAADe 2009, Istanbul, Turkey, 2009, 713–718.

[6] 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

[7] Lin, C.-J.: Design Criteria Modeling - Use of Ontology-Based Algorithmic Modeling to Represent Architectural Design Criteria at the Conceptual Design Stage, Computer-Aided Design and Applications, 13(4), 2016, 549–557. http://dx.doi.org/10.1080/16864360.2015.1131551

[8] Miller, N., LunchBox, http://www.theprovingground.org/.

[9] OpenBuildings, Fuji Kindergarten, http://openbuildings.com/buildings/fuji-kindergarten-profile-2425.

[10] Oxman, R.: Digital Architecture as a Challenge for Design Pedagogy: Theory, Knowledge, Models and Medium, Design Studies, 29(2), 2008, 99–120. http://dx.doi.org/10.1016/j.destud.2007.12.003

[11] Peters, B.: Computation Works: The Building of Algorithmic Thought, Architectural Design, 83(2), 2013, 8–15. http://dx.doi.org/10.1002/ad.1545

[12] Protégé, http://protege.stanford.edu/, Stanford University.

[13] Terzidis, K.: Algorithmic Architecture, Architectural Press, Burlington, MA, 2006.

[14] Tezuka, T., The Best Kindergarten You've Ever Seen, https://http://www.ted.com/talks/takaharu_tezuka_the_best_kindergarten_you_ve_ever_seen.

[15] VisualARQ, http://www.visualarq.com/, Asuni CAD.

[16] Woodbury, R.: Elements of Parametric Design, Routledge, New York, 2010.

[17] Yu, R.; Gero, J.: An Empirical Foundation for Design Patterns in Parametric Design, in: Proceedings of the 20th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA 2015), Daegu, 2015, 551–560.

[18] Yu, R.; Gero, J.; Gu, N.: Architects' Cognitive Behaviour in Parametric Design, International Journal of Architectural Computing, 13(1), 2015, 83–102. http://dx.doi.org/10.1260/1478-0771.13.1.83