COMPUTER-AIDED DESIGN & APPLICATIONS, 2016
VOL. 13, NO. 4, 549–557
http://dx.doi.org/10.1080/16864360.2015.1131551

Design Criteria Modeling - Use of Ontology-Based Algorithmic Modeling to Represent Architectural Design Criteria at the Conceptual Design Stage

Chieh-Jen Lin

Tainan University of Technology, Taiwan

ABSTRACT
This paper proposes an ontology-based parametric modeling tool termed "Design Criteria Modeling (DCM)," which applies a graphic predicate tool and semantic ontologies of architectural topology. DCM aims to help architects in representing, exploring, and validating design criteria by means of parametric 3D model at the early design stage. By applying a semantic ontology reasoner, DCM can help architects to determine whether conceptual models meet the semantic ontology of proposed design criteria.

KEYWORDS
Algorithmic modeling; architectural design criteria; semantic ontology; parametric design

1. Introduction

Due to continuous reductions in the cost of computer graphics and processing power, 3D modeling software has become an essential tool for architectural design education and practice. There are at least three types of 3D modeling tools for different architectural design applications: (1) Building Information Modeling (BIM), (2) rapid 3D modeling, and (3) algorithmic modeling.

Thanks to its use of 3D visualization and database technology, BIM has gradually replaced the 2D drawings of CAD as a means of solving information integration problems among different disciplines within the Architecture-Engineering-Construction (AEC) industry. Based on the Industry Foundation Class (IFC) standard[5], BIM enables sharing of semantic information concerning 3D building components, and thereby facilitates communication among different AEC disciplines. However, the cognitive and learning overload caused by the complexity of BIM applications is still a major obstacle in architectural design. Based on the Levels Of Development (LOD) protocol[4], lower LOD objects, such as massing objects in Autodesk Revit, can reduce the burden of modeling works at the early and conceptual stages by minimizing details. However, since BIM focuses on representing a building product, rather than the design processes, architectural design criteria for generating or validating a building model are usually not easily represented and delivered.

For early and conceptual design, a growing number of architects employ rapid modeling tools, such as SketchUp and Rhino, when freely exploring geometric possibilities, rather than BIM applications that are more useful for detailed modeling of building components. However, rapid 3D models lack semantic information concerning the fundamental components of a building, and thus also make it difficult to determine whether models satisfy proposed design criteria. Some tools can solve some building performance issues. For example, Autodesk Vasari can evaluate sunshine and shadowing of massing models for energy saving in sustainable designs, but other design criteria require different validating algorithms that must be proposed by users.

Since algorithmic modeling tools such as Grasshopper emerged, generating complex forms through the use of algorithms has become much easier than before. The visual programming interfaces of algorithmic modeling tools are easier to learn than textual programming tools, and allow architects to freely explore creative ideas with the help of immediate, visual feedback. Algorithm-based generative modeling has therefore become popular among architects as a means of composing algorithms for generating complex building forms when engaging in geometric creativity during early design stages. On the other hand, proposing algorithms for representing and validating design criteria has become much easier than before. For example, a study has proposed a algorithmic

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

method for finding heuristic solutions for sustainable building designs at the early decision stage[6]. However, except in the case of constructability issues involving complex geometric forms, and abstruse issues for where design criteria can be communicated by generative algorithms, how to compose algorithms in order to meet the requirements of design criteria is still more a test of programming skills than of pure design creativity.

To help architects in representing, exploring, and validating design criteria by means of parametric 3D modeling at an early stage, this paper proposes an ontology-based algorithmic modeling tool termed "Design Criteria Modeling (DCM)," which applies the results of previous projects, including visual ontologies of architectural design[13] and a predicate tool for architectural topology[14] based on OWL of Protégé. The goal of DCM is to integrate architectural design knowledge with algorithm-based parametric design tools for representing, validating, and communicating architectural criteria within algorithmic modeling.

2. The approach of design criteria modeling

In the early and conceptual design stages, architectural design does not consist only of representing and validating known design criteria, but also of learning and developing new criteria. However, not all conceptual design criteria are visible and obvious in 3D models. For examples, minimal space requirements[3], visual field and privacy issues of openings, circulation of indoor and outdoor spaces, and other design issues proposed by architects are different from issues of concern in the engineering and construction disciplines. Traditionally, architects employ other visual media, which usually consist of a series of diagrams, to represent their beliefs and intentions. Since those visual media are separate from 2D drawings and 3D models, their information cannot be easily converted into BIM or other modeling applications for validation. However, with the emergence of algorithm-based generative modeling tools such as Grasshopper, graphic programming interfaces have become a promising means of representing geometric criteria concerning building forms.

Nevertheless, the absence of appropriate annotations sometimes leads to communication difficulties when working with graphic compositions of algorithms, especially when abstract criteria other than geometric features are encountered. Since design information in BIM consists of three types of design information, which are semantic, topological, and geometric[7], architectural design can be regarded as the conversion and processing of these three types of design information. This paper therefore proposes a "geometric-topological-geometric" conversion framework for modeling conceptual design criteria at an early stage of the design process.

2.1. Modeling the semantic ontology of design criteria

Architectural conceptual design always begins with the elaboration of the building plan, site, and other design contexts, which usually consist of textual descriptions of various requirements. The primary design criteria are consequently the semantic elaborations of design objects and their objectives. In BIM, semantic information concerning building components is default and self-evident. However, the semantic criteria proposed by architects may not only represent a consensus concerning the design context, but also the epistemology for how to interpret the design context, and require further validation in order to avoid contradictions. Semantic ontology techniques were therefore incorporated in DCM in order to deal with semantic criteria.

Semantic ontology is a knowledge representation technique used in artificial intelligence to develop semantic webs[1], which can enable search engines to understand users' intentions. Protégé is a popular tool for developing semantic ontologies of domain knowledge[15], and logic reasoners based on semantic web rule language (SWRL), such as FaCT and HermiT built in Protégé, can help knowledge engineers validate and infer implicit knowledge within an ontology. DCM therefore applies a SWRL reasoner as a tool for validating proposed criteria. However, since architectural design must consider various criteria based on heterogeneous information, developing a complete ontology for architectural design needs extremely complicated and laborious works. A modular approach therefore was proposed to allows a flexible integration of the various design information[11]. DCM project also did not attempt to develop a complete ontology of architectural knowledge, but rather provide a rapid mechanism for representing and validating partial ontology of proposed design criteria as a ontological module. For example, assuming that a set of semantic ontology rules is declared a basic feature of a "Good Outdoor Space" of a building as follows: (1) an "Outdoor Space" having a "Good Landscape Element" implies a "Good Landscape" property (Eqn. 1), and (2) an "Outdoor Space" having a "Good Landscape" property implies a "Good Outdoor Space." (Eqn. 2) A semantic logic reasoner can deduce a new assertion that an "Outdoor Space" with "Good Landscape Element" implies a "Good Outdoor Space" (Eqn. 3) based on these two assertions.

$$\text{OutdoorSpace}(?x) \wedge \text{LandscapeElement}(?y) \wedge \text{hasQuality}(?y, \text{“Good”})$$

$$\wedge \text{hasLandscapeElement}(?x, ?y)$$
$$\rightarrow \text{hasLandscapeQuality}(?x, \text{“Good”}) \quad (1)$$

$$\text{OutdoorSpace}(?x) \wedge \text{hasLandscapeQuality}(?x, \text{“Good”})$$
$$\rightarrow \text{hasQuality}(?x, \text{“Good”}) \quad (2)$$

$$\text{OutdoorSpace}(?x) \wedge \text{LandscapeElement}(?y)$$
$$\wedge \text{hasQuality}(?y, \text{“Good”}) \wedge$$
$$\text{hasLandscapeElement}(?x, ?y)$$
$$\rightarrow \text{hasQuality}(?x, \text{“Good”}) \quad (3)$$

Although the semantic logic reasoner cannot automatically deduce solutions that will satisfy the proposed design criteria, it can detect contradictions, and can help architects maintain the consistency of design criteria. For example, the third rule (Eqn. 3) can remind architect to add some good landscape elements to enhance the quality of an outdoor space, such as a courtyard or a forecourt. Therefore, by designating landscape elements of an outdoor space, such as planting, sculptures, ponds, or gardens as a subclasses or an instances of “Good Landscape Element,” a logic reasoner can determine whether an outdoor space is good or not based on those elements.

2.2. Modeling topological relations of design criteria

Topologies are both the mathematical connections of components in BIM and the fundamental definitions of parametric modeling[7]. However, the topologies of BIM applications are usually prior knowledge, which is embedded in the parameters of models for the purpose of fabrication and construction, and cannot be freely redefined or manipulated by architects. Topological relations nevertheless usually also constitute critical information used in validating whether a model satisfies the semantic ontology of proposed design criteria. For example, the landscape elements of an outdoor space imply the elements are within sight. In other words, those landscape elements should be within, adjacent to, nearby, or far away from a site, but not visually obstructed from the site (Eqn. 4).

$$\text{OutdoorSpace}(?x) \wedge \text{LandscapeElement}(?y)$$
$$\wedge ((\text{isEnclosing}(?x, ?y) \vee \text{isAdjacent}(?x, ?y)$$
$$\vee \text{isNearby}(?x, ?y) \vee \text{noObstructBetween}(?x, ?y))$$
$$\rightarrow \text{hasLandscapeElement}(?x, ?y) \quad (4)$$

The topological criteria of design elements contain the mathematical relations of those elements’ geometric features, concerning their coordination and shape. However, not all topological criteria of design elements or objects are visible or obvious. For example, the purpose of arranging landscape elements is usually not only to ensure the quality of an outdoor space, but also to provide visual elements for adjacent indoor spaces. Therefore, an invisible topological criterion between landscape elements and indoor spaces is the vision of “Window.” This rule can be declared as: if “Window” of “Indoor Space” has the “FaceTo” property with a “Good Landscape Element,” this implies that the “Indoor Space” has the “Good Vision” property (Eqn. 5).

$$\text{IndoorSpace}(?s) \wedge \text{Window}(?w)$$
$$\wedge \text{hasWindow}(?s, ?w) \wedge \text{OutdoorSpace}(?x)$$
$$\wedge \text{LandscapeElement}(?y) \wedge \text{hasQuality}(?y, \text{“Good”})$$
$$\wedge \text{isAdjacent}(?s, ?x) \wedge \text{FaceTo}(?w, ?y)$$
$$\rightarrow \text{hasVisionQuality}(?s, \text{“Good”}) \quad (5)$$

Since the “FactTo” property of “Window” usually is invisible or non-obvious in either 2D drawings or 3D models, architects usually employ sketches or diagrams to visualize this kind of topological criterion. However, it is easy to compose an algorithm in Grasshopper to detect this kind of topology. For example, an algorithm can be developed to calculate the angle between the wall of “Window” and the central connecting line of “Window” and “Landscape Element” before the size of window is determined (Fig. 1.a), or to calculate whether the “Landscape Element” is within the vision cone of “Window” from the center of the “Indoor Space” after the size of “Window” is determined (Fig. 1.b).

However, the complexity of detecting simple topologies, such as separated, adjacent, overlapping, and enclosing, will increase linearly with the number of relevant elements, let alone the case of other complex topologies involving multiple objects, such as surrounding, centralizing, clustering, and branching[10]. Before algorithmic modeling tools such as Grasshopper appeared, how to program topological relation detection algorithms was usually more of a challenge of programming skills than design skills for architects. By employing algorithmic components in Grasshopper, simple topologies between two geometries are more easily detected than before. However, as composing algorithms within Grasshopper is usually too complex to be easily recognized by most architects, composing complex or multiple algorithms within a single topological class is usually more convenient for most users. For example, to packaging “isEnclosing,” “isAdjacent,” “isNearby,” and “noObstructBetween” algorithms into a “hasLandscapeElement” class can simplify the representation of topological criteria for the “Outdoor Space” class. However, complex topologies concerning multiple objects,

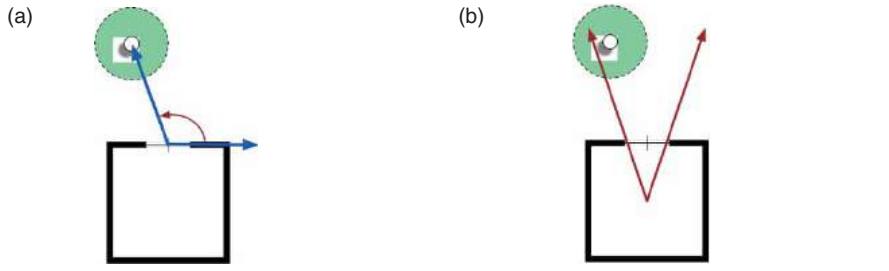

Figure 1. Two Algorithms for detecting "FaceTo" topology of "Window" with "Landscape Element."

which are easily recognized by humans or generated by algorithms, sometimes cannot be efficiently detected or validated using easily-developed algorithms. As a prototype tool, DCM first focuses on providing conversion algorithms for associating simple topologies, which only concern two objects, with semantic relations in the ontologies of design criteria.

2.3. Modeling geometric features of design criteria

Geometric features are the primary parameters of BIM's components, and modeling the detailed geometric features of building components is therefore one of the major tasks performed by BIM applications. In the early design stage, however, design objects usually contain only abstract semantic information, such as building massing, zones, spaces, and other attributes assigned by architects, and their geometric features are usually simplified in order to speed up validation of the topological relations among objects. By applying Grasshopper as an algorithmic modeling tool, DCM is therefore able to use the geometric functions of Rhino for generating geometric features of design objects. However, unlike BIM, the geometric objects of Rhino have no primary semantic information concerning building components, and all generated objects of Grasshopper therefore need to have necessary attached semantic information for retrieving and validating relevant design criteria. Through the help of semantic reasoning, DCM can help users to easily attach semantic information concerning design criteria to generated objects in Grasshopper, which will then trigger semantic reasoning to validate whether relevant design criteria are satisfied or not.

Since generative modeling tools like Grasshopper can input geometric objects in Rhino as parameters for generating 2D shapes or 3D models, DCM therefore also takes geometric features constituting conceptual design objects, such as 2D curves and regions, and 3D massing, as input parameters, and then generates validation results by means of algorithms. For example, the plan shape of a building within a site can be generated from simple design criteria as follows: the "Buildable Area" is equal to the "Shape" of the "Building Site" minus the "Shape" of "Unbuildable Area." (Eqn. 6)

$$\begin{aligned} &\text{Site}(?x) \wedge \text{hasShape}(?x,\ ?g) \\ &\quad \wedge \text{UnbuildableArea}(?y) \wedge \text{hasShape}(?y,\ ?q) \\ &\quad \rightarrow \text{hasBuildableAreaShape}(?x,\ \text{minus}(?g,\ ?q)) \end{aligned} \quad (6)$$

By indicating the subclasses of "Unbuildable Area," such as "Retreat of Building Codes," "Sidewalk," "Existing Building," "Good Landscape Element," et al., and then applying the "Region Difference" algorithm, Grasshopper can therefore easily generate the shape of a possible "Buildable Area" by indicating the "Curves" of the given "Site," and the "Unbuildable Area" within the "Site." However, unlike generative modeling, which is often used to generate complex and sophisticated 3D models, DCM first attempts to model invisible or non-obvious design criteria, such as minimum space requirements, view fields of openings, and geometric constraints of the building codes on design objects, rather than automatically generating optimal or possible solutions. In the "Buildable Area" example mentioned above, the visualization of "Unbuildable Area" and the implementation of "minus" topology is more important than the generated shape of "Buildable Area" when communicating design criteria with others.

2.4. Summary

Based on the foregoing "geometric-topological-geometric" information conversion framework, DCM consequently divides conceptual design tasks into three modeling steps: (1) semantic ontologies of design objects or objectives, (2) topological relations among those objects, and (3) geometric features for visually validating ontologies by means of their topologies. DCM not only aims to provide a tool for incorporating design criteria, especially

those criteria that are invisible or non-obvious in 3D visualizations of parametric models, but also provides guidance for architects when they wish to explicitly represent their design beliefs and intentions.

3. Implementation and initial evaluation of DCM

3.1. Prototype implementation of DCM

One of initial motivations for developing DCM was to improve the knowledge representation abilities of Grasshopper in architectural design, and the initial prototype of DCM was developed within Grasshopper. Since Grasshopper is a DotNET plugin for Rhino, VisualBasic or C# should be the first choice of programming language for developing plugins for Grasshopper. However, the integration of semantic ontology techniques is the biggest difference between DCM and other Grasshopper plugins. Since Protégé was developed in Java, most ontology tools are also available only in Java. Fortunately, the new Rhino 5 also supports the Python script language, and some Python libraries, the such as RDFLib[16] and FuXi[8], and can support the manipulation functions of RDF/OWL and logical reasoning of SWRL, and Jython[12] can support Python script in a Java virtual machine (JVM). The GhPython[9] plugin, which can provide a hook for Python scripts for Grasshopper, was therefore an ideal tool for implementing DCM.

3.2. Initial testing and evaluation

The DCM prototype was tested by students, who were asked to rapidly design an "Architect's Office as a Good Neighbor," which was a topic on Taiwan's architect qualification examination in 2014. The urban context of the site includes middle-rise housing to the east, several low-rise houses to the west, a community park to the south, and a primary school to the north. In addition, the site also contains two old trees and an old two-story dormitory for public servants, and the trees and old dormitory must be retained and reused (Fig. 2).

The design of this project involves determining how the context and content of the site will affect the massing of the office building, and how to arrange outdoor and indoor spaces in the office building so as to be a good neighbor to the community. The context and content of the site not only constrain, but also suggest feasible layouts for the building massing and interior spaces. While these clues may not be obvious, and may even be invisible, learning how to recognize and express the influence of the design context is the major purpose of this kind of exercise. Traditionally, students must first try to develop and

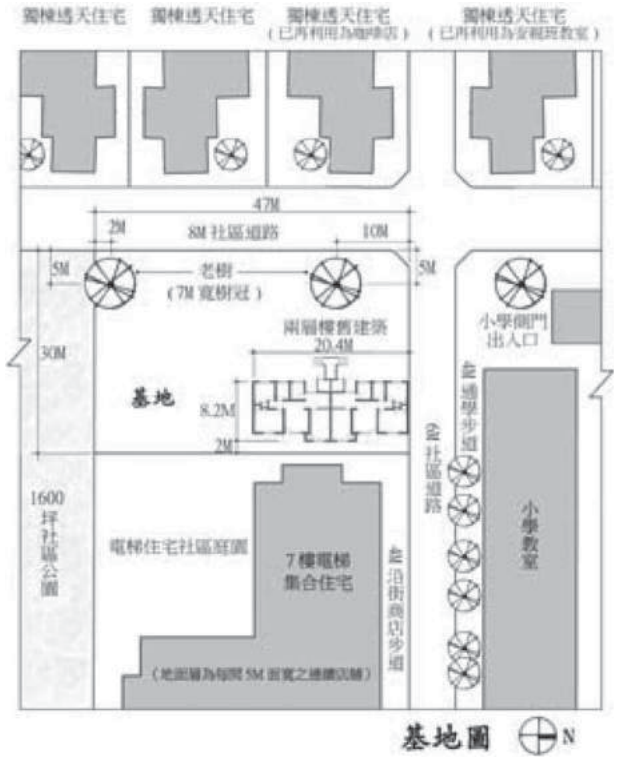

Figure 2. The site context for the rapid design of an "Architect's Office as a Good Neighbor."

express design criteria by means sketches, such as in the case of entrances and access for different types of users (Fig. 3.a), and possible zones of neighborhood activities (Fig. 3.b).

After quick studies in the form of hand-drawing sketches, the students were asked to develop more explicit expressions of their design criteria for conversion to semantic ontology. However, the initial semantic criteria, such as the building codes, traffic, and climate response, recognized by most of the students, usually could not directly shape or generate the building forms without topological or geometric criteria. Experienced students therefore acquired more clues from the site's context, which consisted of further parameters that could be input into criteria for generating the building forms. For example, students realized that the two old trees constituted favorable conditions for shaping outdoor spaces, and could provide scenery and ventilation for indoor spaces. The students were thus able to develop "Good Landscape Element" criteria (Eqn. 1 to 4), and a "Good Vision" of "Indoor Space" criterion (Eqn. 5). Since the building plan required retention and reuse of the old dormitory, "Unbuildable Area" criteria (Eqn. 6) therefore were developed. However, after reviewing all proposed criteria, the students found there was no criterion for "Neighborhood Friendly," and therefore had to develop

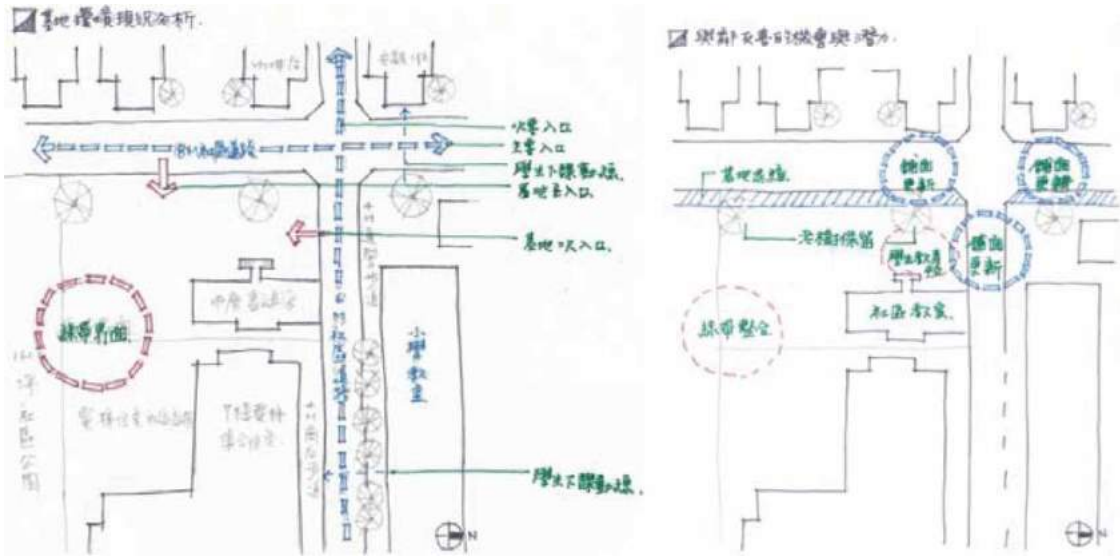

Figure 3. Development of design criteria by means of sketches for an "Architect's Office as a Good Neighbor": (a) Entrances and means of access (b) Neighborhood activities and possible locations.

more topological criteria, such as the rule that "Good Open Space" must be adjacent to "Public Access," and "Public Space," such as a "Community Park." (Eqn. 7)

$$\begin{aligned}&\text{OutdooSpace}(?x)\\&\quad\wedge \text{hasLandscapeQuality}(?x,\ \text{“Good”})\\&\quad\wedge \text{PublicAccess}(?a)\wedge \text{PublicSpace}(?p)\\&\quad\wedge (\text{isAdjacent}(?x,\ ?a)\vee \text{isAdjacent}(?x,\ ?p))\\&\quad\rightarrow \text{OpenSpace}(?x)\wedge \text{hasQuality}(?x,\ \text{“Good”})\end{aligned}\tag{7}$$

This criterion can explain why experienced students chose to place the office building on the southeast side, where the open space around two old trees can be easily accessed by people in order to facilitate neighborhood activities. However, the geometric criteria for generating the open space shape or the building form were still not adequate. More geometric criteria therefore had to be developed, such as that the "Minimum Active Space" around a "Tree" must be larger than 3 meters (Eqn. 8), and the "Minimum Retreat Space" of the new building away from an "Existing Building" must be larger than 2 meters (Eqn. 9).

$$\begin{aligned}&\text{OpenSpace}(?x)\wedge \text{hasShape}(?x,\ ?s)\wedge \text{Tree}(?t)\\&\quad\wedge \text{hasShape}(?t,\ ?v)\wedge \text{isEnclosing}(?s,\ ?v)\\&\quad\wedge \text{isLargerThan}(\text{minus}(?s,\ ?v),\ 3)\\&\quad\rightarrow \text{hasActiveSpace}(?t,\ \text{“Enough”})\end{aligned}\tag{8}$$

$$\begin{aligned}&\text{NewBuilding}(?b)\wedge \text{hasShape}(?b,\ ?s)\\&\quad\wedge \text{ExistingBuilding}(?e)\wedge \text{hasShape}(?e,\ ?r)\\&\quad\wedge \text{isSeparated}(?b,\ ?r)\wedge \text{isLargerThan}(\text{Distance}\\&\quad(?s,\ ?r),\ 2)\rightarrow \text{hasRetreatSpace}(?b\text{“Enough”})\end{aligned}\tag{9}$$

These two criteria can further constrain the open space and buildable area, enabling feasible shapes of the new building plan to be found. However, since the algorithm for detecting minimum space around trees was not so easy for students to implement in Grasshopper, alternatives in the form of "offsetting" the shape of trees (Eqn. 10), and use of the "Existing Building" shape (Eqn. 11) to ensure "Minimum Space," were proposed as follow:

$$\begin{aligned}&\text{OpenSpace}(?x)\wedge \text{hasShape}(?x,\ ?s)\wedge \text{Tree}(?t)\\&\quad\wedge \text{hasShape}(?t,\ ?v)\wedge \text{isEnclosing}(?s,\ ?v)\\&\quad\wedge \text{isEnclosing}(?s,\ \text{offset}(?v,\ 3))\\&\quad\rightarrow \text{hasActiveSpace}(?t,\text{“Enough”})\end{aligned}\tag{10}$$

$$\begin{aligned}&\text{Site}(?x)\wedge \text{hasShape}(?x,\ ?g)\wedge \text{UnbuildableArea}(?y)\\&\quad\wedge \text{hasShape}(?y,\ ?q)\text{ExistingBuilding}(?e)\\&\quad\wedge \text{hasShape}(?e,\ ?r)\rightarrow \text{hasBuildableAreaShape}\\&\quad(?x,\ \text{minus}(?g,\ ?q,\ \text{offset}(?r,\ 2))\end{aligned}\tag{11}$$

These two criteria were easier for student to implement in Grasshopper by applying the 2D/3D-offset algorithm to the geometric shape of trees and existing building. A demonstration is shown as follows of modeling the design criterion of "Minimum Active Space" of two trees, and thus shaping the preliminary building

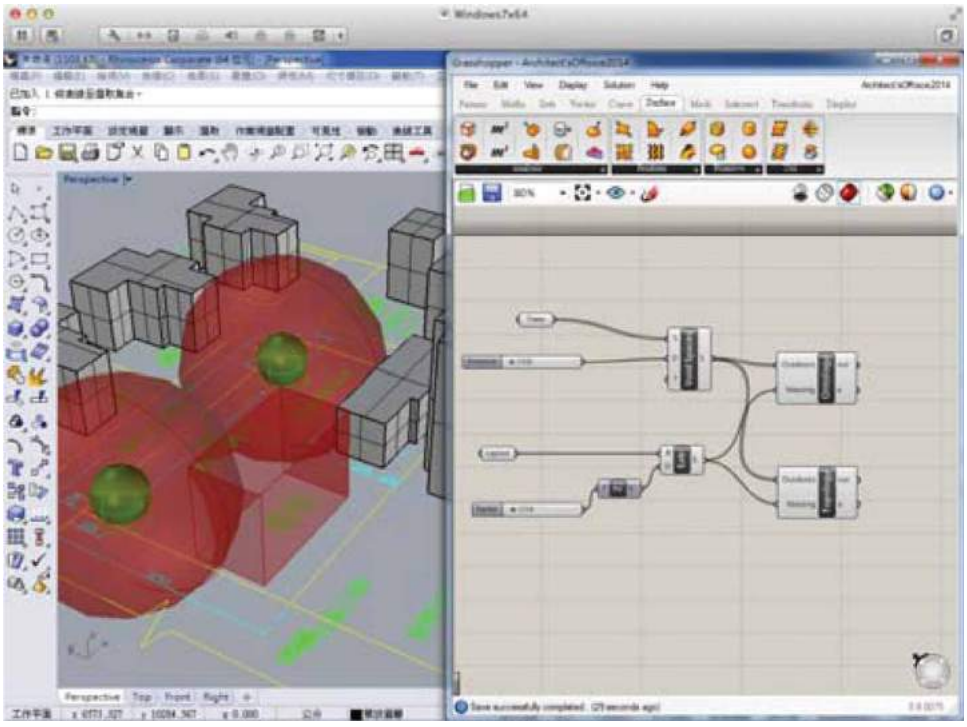

Figure 4. Primary test modeling of design criteria for the site context of "Architect's Office as a Good Neighbor."

massing in Grasshopper based on the "open space" criteria and "buildable area" criteria mentioned above (Fig. 4).

However, it is inevitable that conflicts will occur among different design criteria. For example, since a "Landscape Element" such as the trees located on the west side of the buildable area, a conflict occurs when applying the "FaceTo" criterion of "Window" with "Landscape Element," and the "NotFaceToWest" criterion of "Window" for avoiding the strong sunshine from the west. This was originally a deliberate element of the exam, and intended to challenge students. There are two tactics for solving the conflicting criteria: (1) first rank the importance of the design criteria, then apply design criteria in the order of importance; and (2) search for more criteria which can meet all requirements of the conflicting criteria. The second tactic usually is more satisfactory than the first.

Experienced students recognized the climate consequences of their choices, and therefore took appropriate countermeasures, such as deep balconies, vertical shadings, double walls, plantings, and water pools. This kind of tactic is to declare more criteria for the "FaceToWest" criteria of "Window" in order to solve the conflicts. However, some creative students argued that the vertical shadings usually obstructed the scenery of the landscape elements, and therefore proposed that the old trees could serve as natural vertical shadings to meet this condition. In this case, asking students to model their proposed design criteria not only required novice students to explicitly recognize design problems, and then solve the given problems, but also encouraged creative students to challenge conflicts by proposing appropriate criteria. Through the guidance of DCM, nearby houses, the dormitory building, the old trees, and other contextual objects could become the primary parameters of algorithms for modeling design criteria.

4. Discussion

In order to enhance the utility of generative modeling tools such as Grasshopper in representing and validating design criteria during early architectural stages, this paper proposed a "geometric-topological-geometric" conversion framework for modeling design criteria via 2D/3D visualization. This information-driven approach is discussed as follows.

4.1. Semantic ontology of design criteria

Primary design criteria are usually described using textual information in order to establish reasoning logic concerning how to validate whether a given proposal is satisfied or not. In contrast, while the algorithms in Grasshopper can directly generate geometric results, the logic within the semantic ontologies of design criteria can usually only judge whether a proposal satisfies certain criteria. In the previous architectural exercise, for example, if the two old trees are taken as a "necessary condition" in the ontology, then DCM will declare that the deletion

or relocation of the trees is inconsistent with this criterion. The missing associations between semantic criteria and a proposed design often become a cause of disagreement in both design education and practice. DCM can promote better communication among stakeholders in a project, as in the case of students playing architects and design teachers playing clients and consultants at an architectural school.

By employing a semantic reasoner of SWRL, a semantic ontology of design criteria can be validated in advance, avoiding basic inconsistencies. However, DCM does not implement the full functions of ontology techniques such as those in Protégé, but applies Protégé as an external editor and validating mechanism by exchanging XML files of semantic ontology between DCM and Protégé. The XML file of semantic ontology therefore can be used to share design criteria among different projects. However, since the ontologies of conceptual design are usually partial and incomplete, inconsistencies and contradictions among design criteria are common and inevitable in early architectural design. In the previous architectural exercise, for example, the requirement of western sun shading in response to climate issues, the utilization of the old trees for their scenic effect, and the use of space for neighborhood events reflects contradictions concerning openings in the building massing on the western side. But while dilemmas involving design criteria may cause trouble for AI knowledge engineers, they may also serve as drivers of architectural creativity. When encountering such dilemmas, students can not only try to modify criteria to avoid contradictions, but can also challenge dilemmas by proposing more criteria.

4.2. Topological relationships of design criteria

Converting semantic design criteria into generative algorithms of geometric features is often the first difficulty facing novices applying generative modeling tools to design criteria at an early design stage. Topological relationships among design objects therefore provide clues for the selection of input parameters and composition of algorithms. Using the basic definitions of topological relationships, including "separated," "connective," "adjacent," and "enclosing," DCM can help users to compose more topological relationships associated with the semantic ontologies of design criteria. In the previous architectural exercise, for example, the idea of shaping outdoor spaces using two old trees can be converted into an enclosing topology of the building around the trees, or a separating topology between trees and building as a minimal void for outdoor spaces.

While topological relationships constitute critical information used in validating the semantic ontology of design criteria, it is usually difficult for architects who have insufficient programming skills to implement their ideas using this approach. By applying the algorithms of Grasshopper, DCM can help users to easily validate the design criteria of topological relationships. Although it is possible to apply DCM to automatically generate possible design criteria solutions in Grasshopper, DCM is more useful in the visualization of invisible or non-obvious design criteria than in generating forms by means of algorithms. In the previous architectural exercise, for example, if the two old trees and the community park are modeled as "must-see" scenery from interior spaces, DCM does not need an explanation of why to open western windows facing them.

4.3. Geometric features of design criteria

The ability to easily generate sophisticated forms through the use of algorithms is the biggest advantage of generative modeling tools like Grasshopper. However, abstract architectural design criteria cannot easily be converted into the parameters of algorithmic components in Grasshopper. In contrast, geometric features of design objects are associated with semantic ontology and defined topological algorithms in DCM, and can therefore not only be applied to the representation of design criteria, but can also be the input parameters of algorithms. In the previous architectural exercise, for example, different shapes of objects can be the input parameters of topological algorithms in DCM. Students can input a circle, cylinder, sphere, or other irregular shape as "tree" parameters representing retained criteria, and the validation results obtained by DCM vary with the geometric parameters.

The modeling of design criteria therefore not only involves modeling of the building itself, but also modeling of other invisible and non-obvious contextual design aspects, which are recognized by architects and will shape the building. The geometric features of the design context can affect the expression of design criteria. However, since DCM does not attempt to generate optimal results, the results of validation using DCM will retain enough freedom for geometric creativity.

5. Conclusions

One of major complaints concerning BIM applications is their complexity, which causes cognitive overload and learning difficulties[7]. For conceptual development and schematic design, algorithm-based generative modeling tools such as Grasshopper have become popular among

architects because of their ability to handle complex geometric forms through the simple graphic combination of algorithmic components. However, as the generative algorithms in Grasshopper become more complex, even experienced architects may have difficulty remembering why a specific algorithm has a certain composition, much less be able to explain or communicate their ideas with others. However, since not all issues can be easily represented and discovered by product-oriented 3D models, proposals have been made to attach more informational dimensions to 3D BIM, such as time or phasing (4D), cost (5D), energy performance (6D), and facility management (7D)[2]. Architects will continue to apply 2D sketches and diagrams when studying and solving design issues in the early design stage, and not all architectural design issues require a 3D model for representation or validation. For example, minimum space requirements are more easily recognized in 2D drawings. Furthermore, the effort to integrate more design issues into BIM may be lead to the dilemma of an infinite D's model.

Some design criteria may be too axiomatic, and thus need not be indicated, such as minimum space requirements, sufficient ventilation, daylighting, and ceiling height in a habitable room. Other criteria may be too personal to be integrated, such as the epistemology of design contexts, formal aesthetics, and stylistic preferences. These criteria are inevitably lost in BIM, and may not even be required to be included in BIM. Since 3D models generated using BIM have become a powerful medium of communication in the AEC industry, how to represent and validate proposed criteria, especially those that are invisible or non-obvious in 3D models, and to communicate ideas involving those criteria with other stakeholders and disciplines at an early stage, is still a critical issue for architectural design. The DCM approach described above represents a first effort to apply Grasshopper as a generative modeling tool in modeling design criteria, and can be associated with abstract semantic ontologies and topological relationships. However, how to accelerate and simplify the modeling of complex semantic ontologies and topological algorithms still requires more investigation in the future.

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] Antoniou, G.; Harmelen, F.v.: A Semantic Web Primer, MIT Press, Cambridge, Mass., 2004.
[2] Banks, J., BIM Guilt: How many Ds are you doing?, http://www.shoegnome.com/2012/11/06/bim-guilt-how-many-ds-are-you-doing/.
[3] Banks, J., Minimal Space Requirements and improving your Design Criteria, http://blog.graphisoftus.com/archicad-education/tips-and-tricks/minimal-space-requirements-and-improving-your-design-criteria.
[4] Bedrick, J., 2013 Level of Development (LOD) Specification http://www.bimforum.org/lod, BIMForum.
[5] buildingSMART, Open Standards 101, http://www.buildingsmart.org/standards/technical-vision/open-standards-101/.
[6] Changa, 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), 2015, 310–319. http://dx.doi.org/10.1080/16864360.2014.981460
[7] 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
[8] FuXi, https://code.google.com/p/fuxi/.
[9] GhPython, http://www.food4rhino.com/project/ghpython?ufh.
[10] Ho, H.-Y.; Wang, M.-H.: Meta Form as a Parametric Design Language, in: eCAADe 2009, Istanbul, Turkey, 2009, 713–718.
[11] Hois, J.; Bhatt, M.; Kutz, O.: Modular Ontologies for Architectural Design, in: R. Ferrario, A. Oltramari (Eds.) Frontiers in Artificial Intelligence and Applications, 2009, 66–77. http://dx.doi.org/10.3233/978-1-60750-047-6-66
[12] Jython, http://www.jython.org/.
[13] Lin, C.-J.: Visual Architectural Topology: An Ontology-Based Topological Tool for Use in an Architectural Case Library, Computer-Aided Design and Applications, 10(6), 2013, 929–937. http://dx.doi.org/10.3722/cadaps.2013.929-937
[14] Lin, C.-J.: Architectural Knowledge Modeling: Ontology-Based Modeling of Architectural Topology with the Assistance of an Architectural Case Library, Computer-Aided Design and Applications, 12(4), 2015, 497–506. http://dx.doi.org/10.1080/16864360.2014.997647
[15] Protégé, http://protege.stanford.edu/, Stanford University.
[16] RDFLib, https://github.com/RDFLib.