Dancing Stairs
Object modeling for abstract design concept

June-Hao Hou
Graduate Institute of Architecture, National Chiao Tung University, Taiwan
http://www.arch.nctu.edu.tw/~jhou

Abstract. *This paper reviews empirical studies in product modeling and issues of CAAD, and then proposes a new way of thinking in object modeling for abstract design concepts – by using stairs as the subject of study because of its systematic nature and wide variety of forms. This object model works in the higher level over existing product models and deals with abstract concept only. It provides a mean for capturing design concepts and knowledge. Most stair generator in CAAD are only capable of making regular stairs, let alone those fascinating free-form stairs. The purpose of this research is to find a higher conceptual structure of stairs by the object modeling method, so virtually all types of stairs can be described and modeled. Therefore conceptual design in CAAD would be possible and useful for designers. A prototype on AutoCAD will be implemented to demonstrate how the conceptual structure works.*

Keywords. *Object modeling: abstract design concept; stairs.*

Introduction

Pioneers in the area of computer supported environment claimed that architects need tools which augment their capabilities (Neuckermans, 1992), are in tune with their designerly way of thinking (Cross, 1982), and make smarter use of all computer resources (Schmitt, 1988). Today we have ample storage and computing power, the loaded software and network connection is capable of providing plentiful object library, rich set of modeling tools, ease of data exchange, excellence in graphic output, and many other features. But are we there yet? Can architects and designers express and create freely in CAAD environment that is supposed to be in tune with their way of thinking? If we look into how designers operate the computer, we soon realize the stumble way of thinking when using CAAD.

Since 1990s, researchers have being trying to identify conceptual structures in buildings and hope to model any type of design information or knowledge. Some of the them are: the conceptual model by Neuckermans (1992), Engineering Data Model (Eastman, 1994), generic building product model (Eastman, 1995), information reference model (Luiten et al, 1993), building product data model (Björk, 1995), BAS•CAAD (Ekholm and Fridqvist, 1999), IDEA+ (Hendricx, 2000), IFC (IAI, 2001) and STEP (ISO 10303-1, 1994). CAAD systems can now process and exchange design information by standards. However, all the above approaches focus mainly on construction components and data exchange amongst manufacturers. For them, conceptual modeling is more for conceptualization of data model than capturing the

conceptual ideas in designer's mind. Conceptual (or schematic) design phase in design process is still underdeveloped in terms of CAAD support.

Capturing abstract design concepts

When design, we think in terms of forms and concepts. We then turn to computer and are forced to think in fragmented constructs: points, lines, polygons, extrusion, NURBS surface… and so on. When manipulating the model, we think in polygons, meshes, and surfaces, rather than just the form. Though proficiency in computer may help decreasing the efforts of cognition, design concepts are no longer accessible in computer. Abstract design concepts remain in the designer's mind; the process of destruction and reconstruction of the concept in computer is a one-way course of action. Although product models are intended to capture design information, it is the information of the final product and for construction purpose, not information emerged in design process or in early design phase.

Researches have suggested that the computer not only lacks the ability to recognize design actions, it fails to catch up with the speed designers act and think (Tang and Gero, 2001). Furthermore, in recent studies of product modeling and object-based design, CAAD software tries to predetermine the objects in which the designer is supposed to think, thus leaving the most creative tasks to the programmer and the standardization body like IAI/IFC, not the designer (Turk, 2001). Current conceptual frameworks for CAAD aims mainly at design entities that are familiar by architects (Neuckermans, 1992) and largely depend on geometry as design objects and abstract spaces (Hendricx, 2000). Supports for conceptual design in most CAAD, if not all, are limited. Most of them focus on space objects and constrain management, which is only good for spatial layout and assigning constrains amongst entities.

Object modeling for abstract design concept

Based on the stated problems, we intend to explore a new way of thinking in object modeling for abstract design concepts. It is important for a design support system to support for abstract design concepts, without assumptions about the design method (van Leeuwen et al, 2001). And in Turk's analysis (2001), he pointed out reasons for unsuccessful use of software for design: (1) Representation: Software tends to predetermine the objects in which the designer is supposed to think; (2) Situatedness: When a designer is locked in the small screen real estate and the predefined tool sets, the designer is "thrown" in-a-modal but not in-the-world; (3) Communication: The goal of communication is not to share information but to engage participants into activities; and (4) Particularism vs. holism: Breaking things apart to analyze is a common approach in scientific researches, but design practice is creating a whole product and combining parts in one place. Our goal would be to minimize the vintage problems and go for the right direction.

In common design process (Lang, 1974), conceptual design lies in design and choice phases, and manufacturing and construction in implementation phase. Since the product models are well defined and are purposed for productivity and interoperability, it will be better to leave them intact and put design concept in a higher level architec-

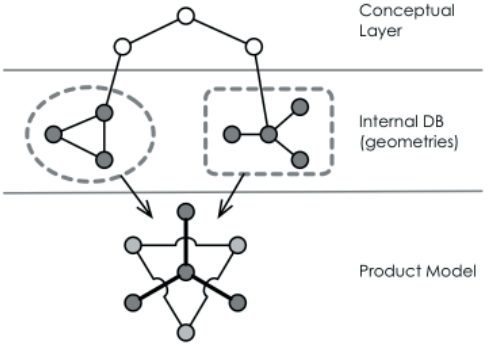

Figure 1. Conceptual layer is the meta-structure of internal DB and product models.

ture. On one hand, it is by itself conceptual, with minimal or no geometric information; on the other hand, it is a meta-structure of product models that capture relationships of design entities. Figure 1 illustrates the proposed structure of object models for abstract design concept (conceptual layer) and the product model layer in a CAAD system.

The object model of design concept, which was an extension of the AIS data model by Sola-Morales (2000), captures relationships amongst Design Entities (DE). Each pattern of relationships forms a concept. However, to really craft the object model of design concept, we need to apply proper abstraction on the well-analyzed relationships. Therefore, the object model could ideally consist of pattern of relationships, and nothing more (Alexander, 1964 & 1979; Moran, 1970). Since the purpose of the model is to describe the relationship, it should be simple and more focusing on the abstract structure of things and not rigorously predetermine all possible building components in the first place. By examining current product models, we found that they strictly adhere to include all possible types of objects. And this brings a lot of usability issues to conceptual design. Let us look at two examples.

Example 1: in most product models, the window is supposed to be placed in an opening on the wall. What about roof window or the window at the joint of the roof and a wall? How do we define it in product model?

Example 2: Should we define the stepped roof of Casa Malaparte as roof or stairs (figure 2)? Apparently it is both (and maybe other roles).

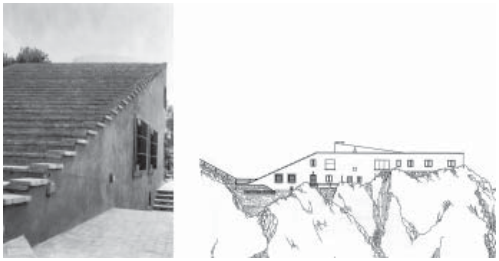

Figure 2. Casa Malaparte (Talamona, 1992)

In the product model, it must be either an instance of stairs or roof, otherwise we have to define a subclass stepped_roof inherited from stairs and/or roof. Thus, the number of classes increases enormously as new classes of construct keep emerging. In the end, we might have to face the dilemma of finding a single class in the haystack or defining our own class from scratch.

Compare to the product model, the object model of design concept is just a set of relationships without embodying their purposes or functions. Follow the above example, the stepped roof is a construct of a series of elevated things and optionally connects to other things on its boundary edges.

If an object model is to be generic, why existing product models leave those special yet famous instances as exceptions?

Interacting with the object model

In addition to capturing abstract concept in the object model, the designer desires to see and manipulate concepts visually and intuitively (Do, 1998). To make the object model of design concept useful, it must goes beyond a data structure and becomes accessible through the user interface. Therefore architects can design by modeled concepts and interact with them. To address the necessity of crafting both the object model and the user interface, we emphasize the simplistic of the model of concept and the user-oriented approach in the process. If we have the model of design concept as data structure only and no visual representation for users to interact with, existing software tools may be too sophisticated since concepts are meant to be simple. On the contrary, if the conceptual design only happens on user interface and all captured data are still destructed into geometric elements, there is no way to back-trace the design concept in the future.

Sketch-based modeling has emerged since late 1990s, such as SketchUp by @Last Software

(www.sketchup.com: May 2005). Designers are able to play with their 3D sketches without worrying the logical construct or object relationships. To some extent, sketch-based CAAD unleashes the power of free form conceptual design. However, lacking relationships amongst concepts has rendered enormous drawback on reusable references and knowledge.

Making stairs

Historically the stairs are critical components providing functions of vertical transportation, connection, linkage, pathway, and many philosophical and inner meanings. Besides the rich characteristics of the stairs, we are most interested in how stairs and the similar components in architecture are designed, structured, and used. One reason is that its structure seems to be pervasive in architecture. The other reason is that, quite interestingly, what stair generators in most CAAD can do is mediocre stairs. They cannot easily model the Spanish Step in Rome, the space-saver stairs, the dancing stairs (winding and curvy shaped), or even the rounded end of the beginning step (bull nose) that is commonly seen in town house. Not to mention the free form stairs by contemporary designers.

Construct of the stairs

Generally speaking, a staircase is a component connecting two slabs in which a number of steps and landings are arranged for transportation purpose. Further conceptualize the stairs by taking similar components into account, such as ramps, extreme cases of stairs, elevated platforms, and even floors, we concluded one fundamental aspect that are universal – they are all elevated structure. Such structure may be self-repetitive and/or grouped with other components as composite, and is referred as Elevated System (or ES). The simplistic form of an ES is just a relationship describes that one Design Entity (DE) is elevated over another. Since a relationship may be a container of one or more DEs or relationships, an ES thus may self-reproduce into many ESs – a recursive structure (Figure 3a).

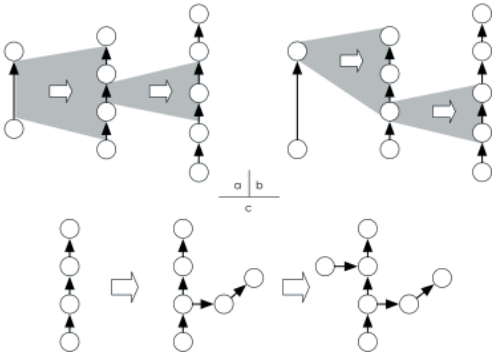

Figure 3. (a) ES develops by expanding relationships; (b) ES develops by expanding DE; (c) ES developed into multi-dimensional linked list.

A Design Element can be anything from single geometry or relationship to a set of them. That is, a DE can be replaced by a composite of DE and relationships (Figure 3b). Data structure wise, the Elevated System is very similar to the multi-dimensional linked list in Computer Science. The only geometric aspect is its gradual elevated development, yet still no specific constrains been set. Furthermore, there may be one or more ES connects to or from a DE – a branch structure (Figure 3c).

In architect's mind

A staircase on a plan drawing is merely a compound symbol indicating its four basic properties: where it begins, where it ends, the line of travel, and the shape (or boundary) of the staircase. Despite their geometric and physical meaning, they represent what architects really have in mind when putting the staircase in the drawing. Speaking of intuitiveness, this is what the architect thinks when sketching.

The top and bottom boundary lines are by far the only information we need to define a staircase. Down to the geometric phase, these two boundary lines define the slope, total length of run, total height, and the stairwell (including head space).

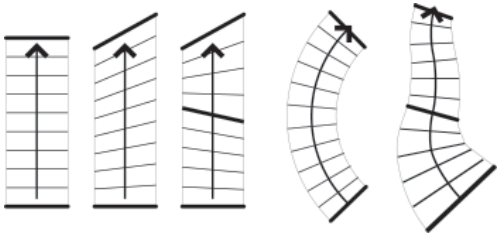

Figure 4. Key Design Entities (thick step lines) control the stair form

Ambiguity or flexibility

The step and stair landing are types of slab. Slab is a DE, and is the super-class of any types of platform – conceptually flat. What is the difference between step, landing, and floor slab? A step is connected by exactly two ESs, one upward, and the other downward. A landing can be connected by more than two ESs. Where other types of slab, such as floor, can be connected by any number of ESs and DEs (openings, walls…). In a T-shape staircase, the landing is connected by three stair flights (ES). Since the landing is also a platform, there may be openings such as door attached to it. In this case, the landing is also a platform. Ambiguity is one of the advantages of the object model of concept, which leads to higher flexibility. What ambiguity means is that the roles of an entity are not predefined in the system, it is to be determined by the designer, or even left undefined.

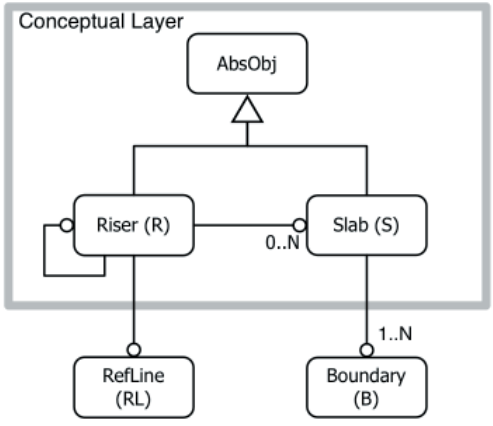

Figure 5. The Object Model of Abstract Design Concept of Stairs

A keyframe approach

Iterative elevations are the logical (or structural) construct of the stairs. Taking the concept of keyframing in animation, development of ES could become more manageable and flexible in many aspects. As a result, custom designed steps can be easily handled by assigning key-DE (key-step). Figure 4 illustrates various situations of using key-entities in stair designs. Thick lines indicate key-entities.

Stair making on computer

Using the built-in stair generator in most CAAD software, one can create stairs in no time by adjusting parameters and hitting buttons, but the freedom to design creatively has been sacrificed. If the designer intends to design stairs that are not possibly done in the stair generator, the only way is to start from scratch – by dealing with points, lines, facets, and tons of modeling tools.

The problem is that those stair generators predetermine stairs as a well-defined structure based on empirical studies, knowledge, and experience at large – mostly come from construction industry. When fitting into this parametric model, the only freedom is to adjust parameters – creativity has been significantly constrained. Some obvious cases that are difficult or impossible to create directly in stair generators have been discussed in previous paragraphs. We simply cannot make variable depth of treads, steps that can be dynamically changed into landings, or steps in irregular shape…etc. Interestingly, the most beautiful and unusual stairs seen in architecture portfolios are mostly beyond the default types of existing stair generators.

A better conceptual model of stairs

The conceptual model is minimal compare to product models. Let along ES and DE, we specialize them into Riser, Slab, Reference Lines, and

Boundary. The preliminary object model of the design concept for stair is:

Based on our reasoning, specialized object model is really not our intent. Higher level of abstraction or conceptualization is the main interests of this research. We use the stairs as the starting point, as the foods for thought, and as the immediate goal of challenge. We try to model every possible type of stairs by the object model of design concept and further elaborate the details of its specifications.

Implementation and expected result

In this research, we claim that during conceptual design phase the CAAD tool should not intervene and aggressively assist or participate with the designer. Instead, the tool should provide an open environment to work with and an object model that is high level of abstraction of the subject. Since the tool is working in high level, it does not conflict with geometry and modeling process of existing CAAD systems. In addition, it serves as the second order construct of the existing tools. So it can be implemented and integrated into most CAAD systems. In this research we choose AutoCAD with Visual LISP and VBA as the platform.

Based on the data model aiming at capturing conceptual architecture information (Sola-Morales, 2001), the proposed object model per se was applicable to wide variety of elevated systems, such as elevated platform, stepped terrain, stairs, ramp, and floors. To resolve the relationships amongst objects, notion of keyframe animation was used as the main construct. Since the only construct is the relationship, every physical components and geometries attached to a relationship can be changed at any time.

That is, except the conceptual structure and its function—vertical transportation, everything in a staircase can be changed. Thus free-form stair design becomes possible on computer. This research discusses a new way of seeing and modeling building components, without worrying too much about geometries compare to current conceptual modeling methods.

Taking the example of the big leap of computer user interface from command line interface to windows-based GUI, the core of the computer does not change at all. It was the way users interact with computer changed. By leaving existing software tools and modeling methods as they are, the proposed conceptual model (i.e. the object model of abstract design concept) and the interface is an attempt to fill out the gap between designer's way of thinking and computer's way of assistantship.

This research is conducted based on empirical studies of stairs and common understanding of how designers perform conceptual design. And for experimental purpose, staircase was taken as the subject of study rather than the whole building. Thus this paper is by no mean a conclusion or the final product of the object model for conceptual design. However, this paper has revealed some possibility beyond traditional researches in object modeling. More works on data collection and user tests are undergoing, a more complete prototype of the interface demonstrating the object model will be done by the end of this year.

Acknowledgement

This research is part of the ongoing doctoral research of the author at Harvard Design School, Cambridge.

References

Alexander, C.: 1964, Notes on the Synthesis of Form, Harvard University Press, Cambridge, MA.

Alexander, C.: 1979, The Timeless Way of Building, Oxford University Press, New York.

Björk, B-C.: 1989, Basic Structure of a Proposed Building Product Model, in Computer-Aided Design, 21(2), pp. 71-78.

Björk, B-C.: 1995, Requirements and information structures for building product data models. Espoo 1995, Technical Research Center of Finland, VTT Publications 245.

Cross, N.: 1982, Designerly ways of knowing, Design Studies, 3 (4), pp. 221-227.

Do, E. Y-L.: 1998, The Right Tool at the Right Time: investigation of freehand drawing as an interface to knowledge based design tools, PhD thesis, http://www.arch.gatech.edu/~ellen/thesis.html

Eastman, C. M.: 1991, Use of Data Modeling in the Conceptual Structuring of Design Problems, in G. Schmitt (ed.), the Proceedings of CAAD futures '91, Zurich, Switzerland, pp. 207-223.

Ekholm, A. and Fridqvist, S.: 1999, The BAS•CAAD information system for design – principles, implementation, and a design scenario, in G. Augenbroe and C. Eastman (eds), proceedings of the CAADfutures'99 Conference Computers in Building, Georgia Institute of Technology, Atlanta,1999.

Hendricx, A.: 2000, A Core Object Model for Architectural Design, PhD Thesis, Dept. Architecture, K.U.Leuven, Heverlee, Belgium.

Heylighen, A. and Neuckermans, H.: 2000, Design(ing) knowledge in architecture, in Proceedings of EAAE/ARCC Conference, Paris, July 2000.

IAI: 2001, International Alliance for Interoperability, <www.iai-international.org>.

ISO 10303-1: 1994, Industrial automation systems and integration Product data representation and exchange – Overview and Fundamental Principles, International Standard, ISO TC184/SC4, 1994.

Lang, J. and Burnette, C.: 1974, A model of the design process, in J. Lang (ed.), Designing for Human Behavior: Architecture and the Behavioral Science.

Van Leeuwen, J., Hendricx, A., Fridqvist, S.: 2001, Towards Dynamic Information Modeling in Architectural Design, in Proceedings of CIB W78 2001, Mpumalanga, South Africa, 2001.

Luiten, G., Troese, T., Björk, B-C. et al: An information reference model for architecture, engineering, and construction, in the Proceedings of the First International Conference on the Management of Information Technology for Construction, Singapore, August 1993

Moran, T. P.: 1970, A Model of a Multilingual Designer, in T. M. Gary and Design Methods Group (eds), Emerging Methods in Environmental Design and Planning, MIT Press, Cambridge, MA, pp. 69-78.

Neuckermans, H.: 1992, A Conceptual Model for CAAD, in Automation in Construction, 1(1), pp. 1-6.

Sola-Morales, P.: 2000, Representation in Architecture: A Data Model for Computer-Aided Architectural Design, DDes Thesis, Harvard Design School, Cambridge, MA.

Schmitt, G.: 1999, Information Architecture: Basis for CAAD and its Future, Basel, Switzerland: Birkhäuser.

Schmitt, G.: 1988, Microcomputer Aided Design for Architects and Designers: Wiley, New York.

Talamona, M.: 1992, Casa Malaparte (V. di Palma trans): Princeton Architecture Press, New York.

Tang, H. and Gero, J.S.: 2001, Roles of knowledge while designing and implications for CAAD systems, in J.S. Gero, S. Chase and M. Rosenman (eds), the Proceedings of CAADRIA '01, Key Centre of Design Computing and Cognition, University of Sydney, pp. 81-89.

Templer, J.A.: 1992, The Staircase: History and Theories, MIT Press, Cambridge.

Turk, Z.: 2001, The Reasons for the Reality Gap in CAAD, in the Proceedings of eCAADe 2001, pp.156-160.