A DIAGRAM-BASED COMPUTER-AIDED DESIGN INTERFACE IN CONCEPTUAL DESIGN

JEN-HUI KUO
Graduate Institute of Architecture, National Chiao Tung University, Hsinchu, 30050, Taiwan
Email Address : kjh@arch.nctu.edu.tw

Abstract. The paper describes a prototype of diagram-based interface (which we call DBI) in conceptual design. We are interested in the interface in visual thinking process. From diagram studies in three areas, we summarize the concepts as the essential utility of DBI. Then we introduce the component of DBI, implementation and mechanism.

1. Introduction

1.1. COMPUTER-AIDED TOOL IN CONCEPTUAL DESIGN

In Conceptual design stage, we usually represent ideas with sketches. Since computer had become an aided design tool, we have two approaches in this field. One is the computational search process (Simon, 1982; Stiny, 1980); another is the visual thinking process between manipulation of tools and visual feedback (Schon and Wiggins, 1992; Gross, 1996; Kurmann, 1998). When we divide design behavior into three parts: information input, information process, and information output (Wang, 1999), each approach focuses on different part.

In the computational search process, it focuses on the information processing. Computer plays a role as a design production system which process information and contains a set of rules, a database and a control strategy (Rich, 1983). Examples are “Palladio grammar” (Stiny and Mitchell, 1978), “Topology-1” (Akiner, 1986), and “Schematic-Designer” (Liu, 1991). In the visual thinking process, it focuses on the information input and output. Information processing is not handled by computer, but by human. In the seeing-moving-seeing process (Schon and Wiggins, 1992), what we concern about are what to manipulate and what to see.

1.2. THE INTERFACE IN THE VISUAL THINKING PROCESS

What to manipulate and what to see are interface problems. In the visual thinking process, the interface of computer-aided tool has its limitation due to the obstacle of manipulation (Gross, 1996; Kurmann, 1998; Won, 2000). For this reason, to have interface to be more convenient is mentioned a lot. Examples are “Sculptor” (Kurmann, 1998) and “BUILD-IT” (Fjeld, Bichsel and Rauterberg, 1998), which are pointing out intuitive interactive ways between human and computer. But these systems discuss the usability of interface and the possibilities of devices.

The paper will not discuss the variables of device, but discuss the usefulness of interface (Landauer, 1995). McGrenere and Ho (2000) describe the usefulness as follows: “ A useful design contains the right functions required for users to perform their jobs efficiently and accomplish their goals. ” That is, we should concern with the utility of interface. However, what utility is necessary for designer to use in conceptual design?

1.3. DIAGRAM-BASED INTERFACE

Diagram is the ways of representation and analysis in design method (Wang, 1999), used to explore ideas and solutions in conceptual design (Do, 1997). From the point diagram plays an important role in design process, we can regard diagram as a necessary utility. Such examples of computer-aided tool are “Constraints” (Gross, Ervin, Anderson and Fleisher, 1988), “CDT” (Dave, 1993), and “Napkin” (Gross, 1996). “Constraints” supports diagram reasoning about designs and designing. “CDT” claims design problem are not well structured to begin with and there is clearly a need for providing computing tools that enable incremental problem structuring, letting users define their graphic syntax in design process. The position is similar with “Constraints”, which wish to provide languages and tools for designers to use. “Napkin” can recognize the dimension and position of constraints by recognizing and parsing of bubble diagram drawn by freehand sketch.

These systems have different definition of diagram. “Constraints” illustrate diagram impose organizational, adjacency, and leaving position, size, and even shape unspecified. “CDT” define diagram as association diagrams, which contain object and relations, but we see (bubble) diagram has dimensions and shape in “Napkin”. For the discussion convenient below, we define diagram as two types: Topology-diagram (T-diagram) and Geometry-diagram (G-diagram). There are objects and relations in both T-diagram and G-diagram. But object in T-diagram has no dimension, and it’s shape is circle. Object in G-diagram has dimension, and it’s shape is limitless.

What is the advantage when we have the definition of T-diagram and G-diagram? Does it help us clarify the character of diagram and essential detail utility? In order to answer the question, we will review discussions of diagram in three areas (design method, graph thinking and architectural design practicing field, sketch study), trying to find out some results, which can be described in our interface.

2. Diagram in related areas

2.1. DIAGRAM IN DESIGN METHOD

Wang (1999) think diagram is an abstract graph consisting of object and channel, presenting the relations and attributes of object. When using diagram as a functional diagram, objects have no details but with functions and attributes. Joedicke (1987) also think diagram can be used to study the relationships of spaces, and diagram does not concern with dimensions, but with the topology. Therefore we know their discussions of diagram belong to T-diagram.

2.2. DIAGRAM IN GRAPH THINKING AND ARCHITECTURAL PRACTICING FIELD

Van Berkel and Bos (1999) think the essence of diagrammatic technique is not limited to a liner logic and is a visual tool of information compression. The character of information compression let a graph have many meanings. Laseau (2001) provide an example: in graph analysis, line is two-ways which means when kitchen connects to living room, we can also say living room connects to kitchen. Beside this, Laseau provide a syntax rule with graphic analysis. He treats objects as circle, relations as line, variations of circle and line as modification. Although Van Berkel and Bos have no definite definition of diagram, Laseau's opinion points to T-diagram.

2.3. DIAGRAM IN SKETCH

In sketch study, Suwa and Tverskey (1997) point out that designers have "focus shift" activities in segments in design activity. On their encode categories of segments, we find some categories relating to diagram, e.g. local spatial relation, space, shape, and size. These categories are similar to our definition of diagram. For example, we can treat spatial relation and space as a T-diagram, and treat shape and size as a G-diagram. When we apply the result of "focus shift" as our reference of interface, we can interpret the "focus shift" concept to that designer have "focus shift" activities between T-diagram and G-diagram.

2.4. INTERFACE SUGGESTION

From section 2.1 and section 2.2, we know objects in T-diagram contain attributes and can be manipulate in syntax rule. In section 2.3, we know designers have "focus-shift" activities between T-diagram and G-diagram. That is, sometimes designer do something in T-diagram, and sometimes in G-diagram. To our interface, the point has a suggestion of two windows, in which T-diagram exists in one window and G-diagram in another. We can also find the two windows concept in VR Sketchpad (Do, 2001), which convert diagram to VR environment. But that is one-way process. In designer's "focus shift" process, our interface must allow designer have two-ways process.

3. DBI Prototype

3.1. POSITION

From the suggestion in section 2.4, we have two parts in DBI: T-diagram environment and G-diagram environment. For the reason of that a design exist in both environments, each situation of the two-diagram environments has influence on another. As section 2.1 point out that T-diagram is like the functional diagram, which is used in program stage. G-diagram environment deals with the characters of dimension and shape as the basis of further form generating (Figure 1).

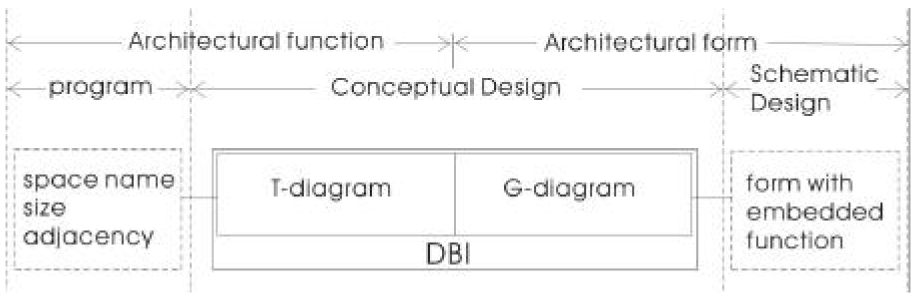

Figure 1. DBI in conceptual design

3.2. MANIPULATION

From section 2.1 and 2.2, we have two elements in T-diagram environment: object and relation. In our definition of T-diagram in section 1.3, object does not have dimension, but store the information of dimension. For the convenience of information visualization, we let object have the freedom of size changing according the size information in it (Figure 3). There are three properties in object: solid or void, geometry, and size. We represent object as signal. Relation has three kinds: adjacency, not-adjacency and not define. We represent relation as action (Figure 2).

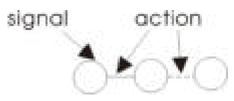

Figure 2. Signal and Action

In D-diagram environment, the information of shape is mapping from signal in T-diagram environment, so we can see the information (e. g., dimension, location) of the shape. In other words, when program in T-diagram is changed, the shape in G-diagram would be changed in synchronism (Figure 3).

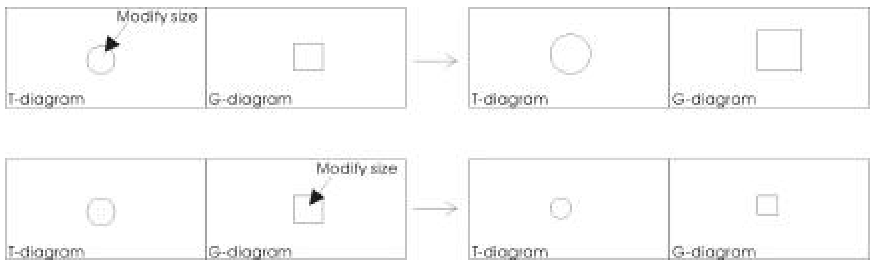

Figure 3. Modifying size of a signal in each environment

Figure 3 describes a situation when there is a signal. In addition to the information in the signal, we can modify the situation of action when there are more than two signals (Figure 4).

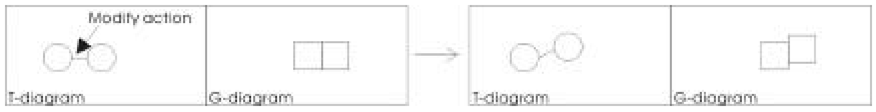

Figure 4. Modifying the action

3.3. IMPLEMENTATION

Composition and decomposition are often used to solve problems in architecture. Flemming and Chien (1995) handle spatial hierarchy by top-down process. In figure 5, if we treat Day Zone as a fixed geometric rectangle (shape, size are fixed), and then add three function: Dayroom, Dining room and Kitchen, we can say the process is similar to "function follows form".

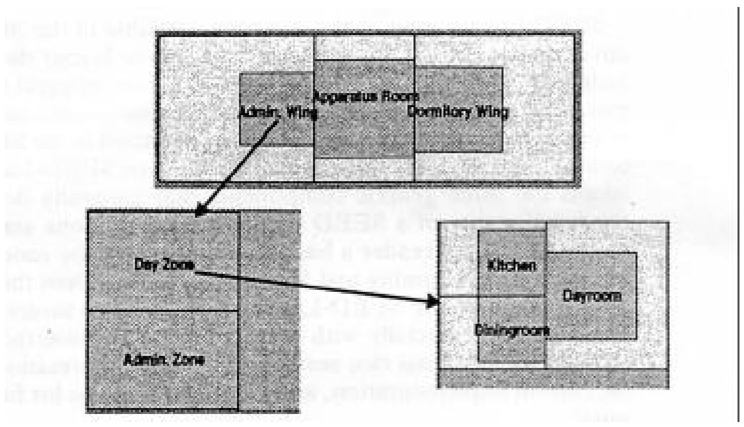

Figure 5. Top-Down Layout Design through Three Abstraction Levels (after (Flemming and Chien, 1995))

Expanding the concept, we can group some signals for the reason of "function follows form". In figure 6, the total volume of the group is fixed, when the volume of one signal is creasing, another is decreasing. But when the two signals are isolate, the volume changing of one signal will not change the volume of another.

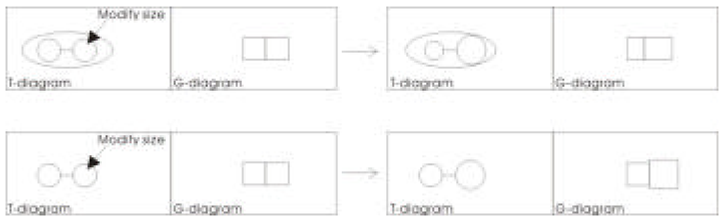

Figure 6. Modifying size of one signal in two situations: group and isolation

3.4. INTERFACE

We are working on the prototype of interface, which have three main parts: add and erase, property, and modify. Two-diagrams environment allow input and edit (Figure7).

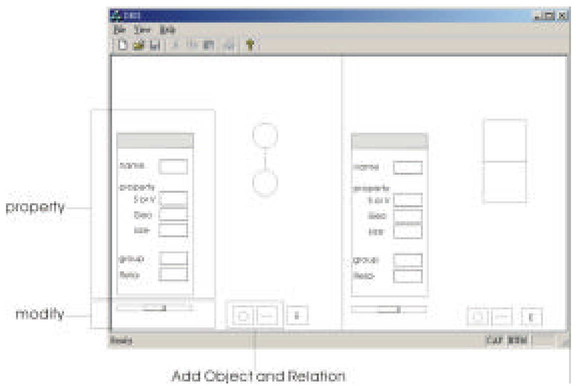

Figure 7. T-diagram window (left) and G-diagram window (right)

As discussions above, we represent object as signal {(name), (group), (solid or void), (geometry), (size)}, relation as action {adj, no-adj, no}. Our function of modify can modify the property of signal, and also can modify the action, just like adjective and adverb (Figure 8).

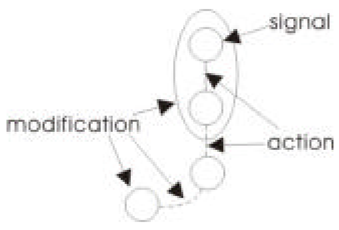

Figure 8. Signal, Action, and Modification

Unlike shape grammar (Stiny, 1980), a computational shape automation system, which begins with an initial shape and applies shape rules to generate shapes. We stand on the similar position with "Constraints" and "CDT", and wish DBI be used for designers to construct their vocabulary and grammar that can avoid critics from semantic meaningless of shape grammar (Wang, 1991).

4. Mechanism

4.1. SYNCHRONISM

The synchronism in Figure 3 is discussed here. Figure 9 shows object class has information (property of signal) and method (e.g., draw()) for signals to inherit. The trigger-event process is two-ways.

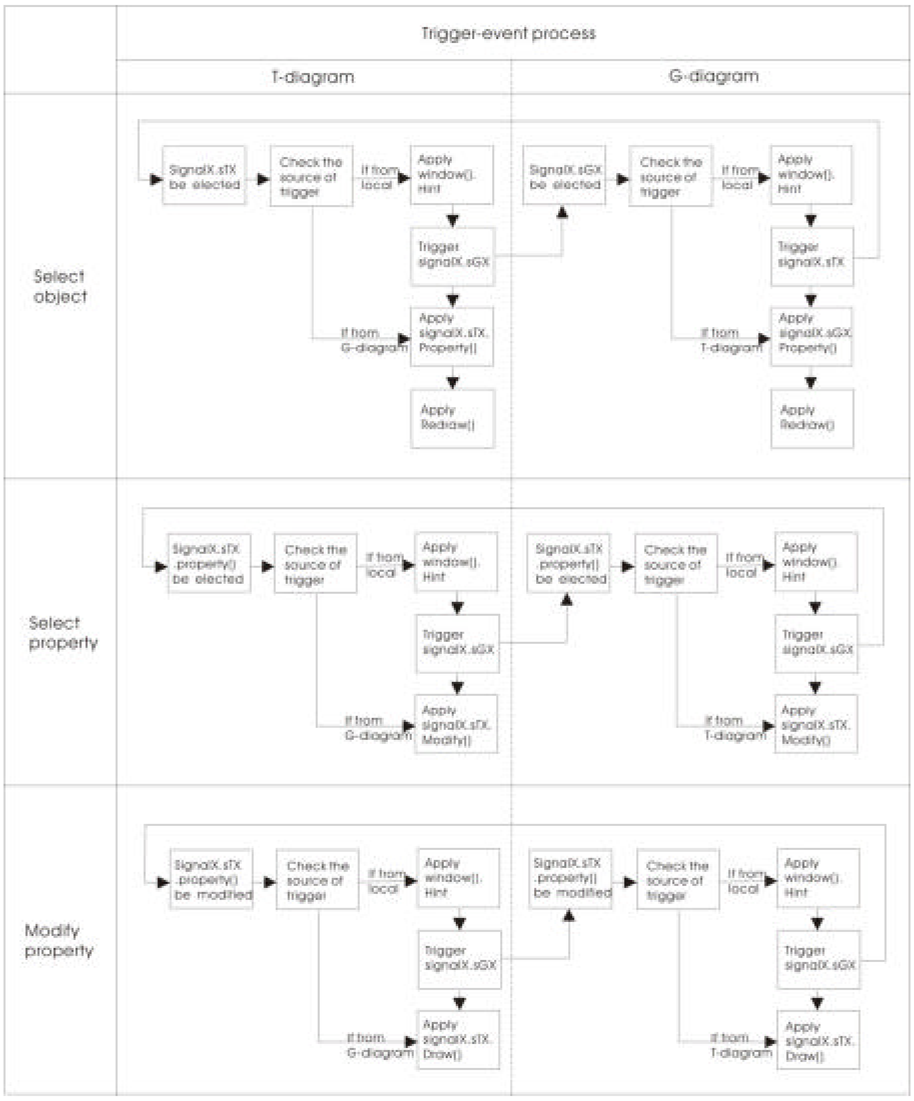

Figure 9. Mechanism of synchronism between T-diagram and G-diagram

4.2. ALGORITHM

In this part, we discuss grouping, and choose Figure 6 as an example. Signals are two rectangles and each rectangle has it's coordinate: ((left, top), (right, bottom)) which used for draw() method in object class. In this case, the volume of signalB increases by V, and then the volume of signalA decreases by V (Figure 10). The algorithm is as follows:

```
If signalB be modified (return V)
then {
signalB.draw((x1-(V/y)), y), (x2, 0))
check the group of object (return signalA)
  signalA.draw((0, y), ((x1-(V/y)), 0)) }
```

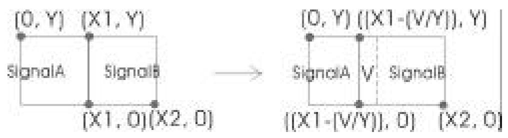

Figure 10. Modifying size of signalB in group situation

5. Discussion and Future Work

We have described the prototype DBI, which is still at early stage of development. It provides a visual environment for users to manipulate signals and actions in conceptual design. It allows designers to do "focus-shift" activities between the two-windows environment. The paper only illustrates the situations of one signal and two signals. In future work, we will discuss complexity situations when more than two signals. Then we will deals more with the issues of "function follows form" and "form follows function".

Acknowledgements

The author is indebted to Dr. T. W. Chang, Dr. Y. T. Liu, and Dr. H. H. Tang for their valuable critiques in ideas and methodology.

References

Akiner, V. T.: 1986, Topology-1: a knowledge-based system for reasoning about objects and spaces, *Design Studies*, **7**(2), 94-105.

Dave, B.: 1993, CDT: A Computer Assisted Diagramming Tool, *in* U. Flemming and S. van Wyk (eds), *CAAD Futures '93*, North-Holland, pp. 91-109.

Do, E. Y. –L.: 1997, Computability of Design Diagrams, *in* R. Junge (ed), *CAAD Future 1997*, pp. 171-176.

Do, E. Y. -L.: 2001, VR Sketchpad: Create Instant 3D Worlds by Sketching on a Transparent Window, *CAADFuture 2001 Proceedings*, PP161-172.

Fjeld, M., Bichsel, M. and Rauterberg, M.: 1998, BUILD-IT: An Intuitive Design Tool Based on Direct Object Manipulation, *in* L. Wachsmut and M. Frölich (eds), *Gesture and Sign Language in Human-Computer Interaction. Lecture Notes in Artificial Intelligence*, Vol. 1371, Springer-Verlag, pp. 297-308.

Flemming, U. and Chien, S. -F.: 1995, Schematic Layout Design in SEED Environment, *Journal of Architectural Engineering*, **1**(4), 162-169.

Gross, M. D., Ervin, S. M., Anderson J. A. and Fleisher A.: 1988, Constraints: Knowledge representation in design, *Design Studies*, **9**(3), 133-143.

Gross, M. D.: 1996, The Electronic Cocktail Napkin-a computational environment for working with design diagrams, *Design Studies*, **17**, 53-69.

Joedicke, J.: 1987, *Design Methodology in Architecture*, Dan-Chin.

Kurmann, D.: 1998, Sculptor - How to Design Space, *CAADRIA '98 Proceedings*, Osaka, Japan. PP317-325.

Landauer, T. K.: 1995, *The Trouble with Computers: Usefulness, Usability, and Productivity*, Cambridge, MA: The MIT Press.

Laseau, P.: 2001, *Graphic Thinking for Architects & Designers third edition*, John Wiley & Sons.

Liu, Y. T.: 1991, Schematic-Designer: a knowledge-based CAD system for schematic design in architecture, *Design Studies*, **12**(3), 151-167.

McGrenere, J. and Ho, W.: 2000, Affordances: Clarifying and Evolving a Concept, *Proceedings Graphics Interface*, PP179-186.

Rich, E.: 1983, *Artificial intelligence*, New York: McGraw-Hill.

Schon, D. A. and Wiggins, G.: 1992, Kinds of seeing and their functions in designing, *Design Studies*, **13**(2), 135-156.

Simon, H. A.: 1982, *The Sciences of the Artificial*, Cambridge, MA: The MIT Press.

Stiny, G. and Mitchell, W. J.: 1978, The Palladian grammars, *Environment and Planning B: Planning and Design*, **5**, 5-18.

Stiny, G.: 1980, Introduction to shape and shape grammars, *Environment and Planning B*, **7**, 343-351.

Suwa, M. and Tversky B.: 1997, What do architects and students perceive in their design sketches? A protocol analysis, *Design Studies*, **18**, 385-403.

Van Berkel, B. and Bos, C.: 1999, *Techniques*, UN Studio&Goose Press.

Wang, G. T.: 1999, *Design Methodology in Architecture 3th ed.*, Tai-Long.

Wang, M. H.: 1991, What Do Architectural Linguists Say? A Critical Review of Languages of Design, *Journal of Architecture, A.I.R.O.C.*, 143-153.

Won, P. -H.: 2000, The comparison between visual thinking using computer and conventional media in the concept generation stages of design, *Automation in Construction*, **10**(3), 319-325.