Representing Geometric Knowledge in Architectural Design

Shun Watanabe

Institute of Socio-Economic Planning
University of Tsukuba
1-1-1 Tennodai Tsukuba-shi Ibaraki
JAPAN

Geometric knowledge is essential to architectural design. A new method of geometric computing is proposed in this paper, in which geometric decision making in architectural design process can be represented naturally. I begin by discussing characteristics of geometry in the architectural design process and making clear the issues in the graphic libraries which have been used as a substitute in current CAD/CG systems. Then, I mention how to represent the spatial geometric primitives and operations from the view of knowledge in algebraic geometry. It is not sufficient for the architectural design process to represent geometric primitives and operations, and a model of geometric decision making is presented to manage the dynamism of the design proceess. I also explain the interaction between the presented geometric decision model and the architectural model which was proposed as the 1ramework to develop our knowledge-based computer-aided architectural design system.

Keywords: design process, geometry, decision making, CAD, AI

1 Introduction

Nature of design is generally considered as the decision making process. Hence, representing various decisions made in the design process naturally is essential to developing knowledge-based computer-aided design systems. Currently, based on the method of Object-Oriented Analysis which has been familiar in the field of computer science, many important studies have been made on describing the designing artifact from the view of its components and their relations. They usually mention not only the representation of shape for visible elements, but also the representation of other important attributes for intensive performance. But their geometric model usually follows the methodology used in current CAD/CG systems. That is, even in these researches, every geometric decision produced in the design process was reduced to the individual graphic object represented by static coordinate values.

AKM(Architectural Knowledge-representation Model) was proposed as the framework to develop our knowledge-based computer -aided architectural design system in 1989. In AKM, the constraints between architectural design components, such as floors, rooms, columns, and beams, were mainly concerned from the view of cognitive science. The graphic objects which express geometric shapes of design components is primary concerned in current CAD/CC systems. In AKM, they can not be stored permanently in their representations, but ca be produced from geometric constraints according to the designer's request of visualization. Graphic objects for architectural components which are fixed to particular coordinate values were considered to prevent designers from changing their design dynamically. Instead, objects which indicate reference lines were introduced for horizontal order, and objects to express floors were substituted for vertical order. Architectural design tends to respect spatial harmony higher than other engineering designs, and geometric order is sometimes given priority over functional requirements

produced from design specifications in the stage of arranging components. Representable design composed of structural elements which must be supported by binomial reference lines is bounded in such a small area.

On the other hand, formula manipulation researches in the field of artificial intelligence result in the current computer math systems. Using these systems, some geometric objects can be dealt as mathematical formula without reducing them to actual coordinate values. However, mathematical formula by itself neither link to geometric meaning directly nor indicate particular geometric objects in CAD/CG systems. What is expected for future design system is not ordinal graphic models, but a geometric model which is able to represent 'geometric knowledge' and also able to cooperate to knowledge-based architectural models.

2 Geometry in design process

Geometric knowledge is essential to architectural design. There have been many researches to analyze geometric characteristics implied in architectural design. What was indicated in these researches is that architectural design is explained by the predicate of geometric facts(orders), e.g., parallel, intersected, and symmetric, which can be found in the composition of architectural components. Currently, not only researches on the characteristics of architectural shape as facts, but also researches on design manipulation by possible grammar in which architectural shape can be translated, have been examined.

On the other hand, every geometric object used in current CAD/CG systems is primary defined by vertices. First, we assume that geometric space can exist in static structures of 2D/3D points, i.e., P(x, y) or P(x, y, z), and geometric objects can also be expressed by mere collections of vertices. A line is defined by two points of origin and destination, and a plane is defined by more than three points. Strictly speaking, they are not lines and planes in mathematical definition, but segments and domains.

The designer's decision made in the architectural design process may be that a certain plane is parallel to another plane, and not individual collections of their actual vertices. The important decision my also be that a reference point for the design is produced as a intersection of reference lines, and not be the coordinate values of the reference point as the consequence of that production. Most geometric objects in an architectural design are produced by previously defined objects and geometric procedures. The consequences of these productions only represent a static state(form) of the artifact, not design decisions themselves. To follow the designer's decision making process, what we must store in intelligent CAAD systems is the designer's decisions expressed as the procedures which imply the geometric facts, restrictions, and contexts.

Geometric decisions are sometimes stored supplementary as the command sequence in the current CAD/CG framework. The process of design decision seems to be expressed in this method, but the command sequence only guarantees to go back to a previous state of design. It is normally found in geometric manipulations that the change of a certain geometric decision propagates other related geometric objects through their restrictions. But this kind of manipulation can't be realized with the command sequence. For example, it is a natural revision in the design process that a certain line produced as the intersection of two planes will be influenced by any movement of each plane. Identically, a certain point defined as the intersection of two lines will be changed properly by any movement of either line.

The hierarchy of facts that geometric objects have been defined in the context of the design process can be called as the representation of decision making in the real meaning of architectural design. However, the structure of decision making has not been the subject in the current methodology of computing geometry. Alternatively, we have followed the geometric representation in which only the actual coordinate values translated as consequences by geometric procedures are stored. This is because most computer languages have introduced the paradigm that procedures and data are explicitly separated.

In some computer languages for the research of artificial intelligence, e.g., LISP and Smalltalk, procedures can be treated as data. That is, in these languages we can assign particular procedures to variables and evaluate them to produce actual values dynamically in any time. The graphic model expected for visual representation is one thing in which static shapes of artifacts are expressed efficiently, while the geometric model expected for architectural design process could be another in which design decisions are expressed and stored as procedures based on these paradigm.

According to the above discussion, I propose GDL(Geometric Decision Library). GDL is designed to express an architectural designers geometric decision making process naturally, and has been implemented on Smalltalk environment. Every statement of geometric objects and procedures presented in the following discussion complies with Smalltalk grammar and can be executed in its implementation. It mainly consists of two super classes, GeometricPrimitive and GeometricDecision. In general graphic libraries, GeometricPrimitive is usually defined by a collection of vertices, that is, graphic libraries are mainly concerned with the visual representation. But in the geometric library, each GeometricPrimitive is expressed by a proper mathematical definition, that is, the geometric library is mainly concerned with geometric knowledge.

3 Representation of geometric primitives

Methodology in which spatial geometric objects are represented and stored by reducing a collection of mathematical coordinate values should guarantee a very simple manner that any complex operation is basically performed to the triplet of numbers (x, y, z). This approach is effective to represent the shape of clearly bounded artifacts and to apply extensionally a geometric transformation expressed as a matrix of 4 by 4. However, it might be difficult in this model to express geometry as restrictions to represent spatial order', which is essential to architectural design process. Production from decision making in design process here is neither a plane nor a line as an architectural substance. The order of design does not mean a particular shape of a certain design element, whereas the data structure of a substitutive geometric object is dissociated from the original meaning intended in the design.

In GDL, spatial geometric objects which are essential to express geometric restrictions in architectural design process(GeometricPrimitives) are described from the stand point of their mathematical nature without limiting to the collection of coordinate values. In mathematics, vector V is generally used to represent space, and it goes without saying that a primitive object in space like a line or a plane, is necessary to express Neometric order of architectural design, which can be written in a mathematical formula. However, vector space is simply employed as the spatial model, and this employment does not mean that all geometric objects are reduced to a collection of position vectors(coordinate values).

Here, a spatial vector is represented with the form of x @ y @ z, and an evaluation of the following statement to indicate a spatial line which passes by two position vectors;

```
Line at: 1 @ 0 @ -1 and: 0 @ 1 @ 1                    (1)
```

produces an instance of Line as its return value;

$$(x - 1) / -1 = (y) / 1 = (z + 1) / 2 \quad (2)$$

Symbols which begin with a capital letter(Line) mean classes in an object oriented paradigm, symbols which include colons(at:and:) represent messages complied with the usage of Smalltalk, and formula expressions as (2) are determined by their intensive method(printOn:). A pair of a position vector and a direction vector is another definition of a line;

```
Line at: 2 @ 4 @ 6 direction: -3 @ 7 @ 0              (3)
```

and an evaluation of the above statement results in;

$$z = 6, (x - 2) / -3 = (y - 4) / 7 \quad (4)$$

In each case, the necessary and sufficient condition to specify a line is a position and a direction of two vectors, which are sometimes translated in their internal procedures. In the previous case (1), it is clear that we simply translate it to;

```
Line at: point1 direction: point2 - point1            (5)
```

The two statements above are elementary examples to define lines by means of vectors, and many other methods are also available to produce lines from various geometric objects. But in every method, two vectors are essential to specify a line as an unique spatial object. In GDL, a line is defined as an object which stores internally a pair of a direction vector and a position vector.

Evaluation of the following statement, to indicate a spatial plane which is defined by a position vector and a normal vector:

Plane at: 3 @ 6 @ 4 normal: 2 @ -4 @ 3 (6)

produce a following instance of Plane as its return value:

$2x - 4y + 3z + 6 = 0$ (7)

In current CAD/CC systems, a plane is always defined by a collection of more than three points, and a similar example to define a plane is also available in GDL. In the following statement:

Plane at: 1 @ 0 @ 0 and: 0 @ 1 @ 0 and: 0 @ 0 @ 1 (8)

three points will be translated into a proper direction vector and a constant value, and produce an instance of Plane as:

$x + y + z - 1 = 0$ (9)

where the instance of Plane (9) no longer stores a collection of points prepared in (8), in opposition to the representation of a plane in current CAD/CC systems.

Representing a plane with a collection of points (position vectors) sometimes causes a conflict to specify a unique plane in space. For example, four individual positioned vectors are not always placed on a same plane. The above definition of a plane is more accurate than the usual definition from a mathematical point of view, and is also useful to perform various geometric operations which will be mentioned in the next chapter. In GDL, a plane is defined as an object which stores internally a pair of a normal vector and a constant value. In the case of representing a explicit domain by a collection of points, an usual face(plane) object should be used instead of the presented plane object.

In general surface modeling, a solid of revolution like a sphere or a cylinder is divided into a certain number of faces and translated into an approximate polyhedron. This is also the sufficient representation of shapes for visualization, but geometric nature of the object disappears as the result. Many primitives are sometimes prepared as solids for ray trace rendering, but are not intended to express geometric meaning. In GDL, a sphere can be produced by a position vector and its radius:

Sphere center: 1 @ -1 @ 0 radius: 1 (10)

and an evaluation of the above statement is:

$(x - 1)^2 + (y + 1)^2 + z^2 = 1^2$ (11)

Many other methods are also available to produce spheres, but the necessary and sufficient condition to specify a sphere is to simply store a position vector and its radius as indicated in (10).

A segment and an arc are ordinal primitives which are usually used as a line and an arc in CAD/CC systems. Defining a geometric object from a view of vertices is also available in GDL. For example;

Segment from: 0 @ 0 @ 0 to: 1 @ 1 @ 1 (12)

The evaluation of an above statement is an instance of Segment, and:

Arc from: 1 @ 0 @ 0 to: 0 @ 1 @ 1 center: 0 @ 0 @ 0 (13)

will instantiate an arc object indicated by given position vectors. They are equivalent to the one in usual graphic libraries, and can also store a collection of vertices (position vectors) internally.

From what has been said above, you can find that mathematical definitions are simply reflected in the internal structure of geometric primitives in GDL. These representations can be adopted into a small number of geometric objects which are welldefined mathematically. It may be indicated that the degree of freedom for architectural shape will be restricted. However, representation of shape should be served by graphic libraries, and what is expected in the geometric library is the representation for geometric order and knowledge.

4 Operation of geometric primitives

Spatial geometric objects which constitute architectural design will be produced as the results of various geometric operations. In current CAD/CG systems, their actual procedures are prepared as a command library, and be fired according to the designers requests which ever they can be instructed from the command line or icon. But under the influence of internal representations for geometric objects, their logic usually employs the methodology in which 'calculation of individual vertices is considered to geometric operation. That is, a geometric operation does not produce a new geometric object as its result, but produces corresponding vertices. In the geometric library, geometric primitives should be managed abstractly, and every geometric operations should be based on the same way in which we solve geometric equations.

In architectural design process, we usually suppose a new plane which is translated by certain distance from a previously defined plane. In GDL, this operation can be expressed with temporal variables named plane l and plane2 as follows;

```
| plane1 plane2 |

plane1 := Plane at: 3 @ 6 @ 4 normal: 2 @ -4 @ 3.                    (14)

plane2 := plane1 translatedBy: 10. and plane2 will be instantiated as:

2x - 4y + 3z - 47.8516 = 0                                           (15)
```

What is important here is that planel was generated to present its mathematical formula, and does not store a position vector(3 @ 6 @ 4) which was the instruction of its production. Therefore plane2 can't be produced by the translation of a particular point of plane l. A new plane can be identified by an existing line and a point, and this operation will be:

```
| point1 line1 plane1 | point1 := 0 @ -1 @ 1.

line1 := Line at: 0 @ 0 @ 0 direction: 1 @ 1 @ 1.                    (16)

plane1 := line1 interconnectionWith: point1. and instantiate plane1 as follows.
2x - y - z = 0                                                       (17)
```

The orbit of an architectural column may be a line decided by two reference planes which indicate the orders of columns, and the procedure to get a line as the intersection of two planes can be expressed with temporal variables named plane l, plane2, and line l as follows:

```
| plane1 plane2 line1 | plane1 := Plane at: 0 @ 1 @ 0 normal: 1 @ 1 @ 1.

plane2 := Plane at: 1 @ 0 @ 0 normal: 2 @ -1 @ 1.                    (18)

line1 := plane1 intersectionWith: plane2. and line1 will be instantiated as;
(x) / 2 = (y + (1/2)) / 1 = (z - (3/2)) / -3                         (19)
```

Its internal procedure is to solve simultaneous equations. To be exact, these equations can be solved to divide the conditions according to the outer product of normal vectors of two planes. Many other operations of geometric primitives are also described in GDL, but as I mentioned in this chapter, every operation is simply expressed from the stand point of geometric knowledge we have studied to solve geometric problems.

5 Model of geometric decision making

To manage the dynamism of the design process, it is not sufficient to represent spatial geometric primitives and operate them in mathematical formulas. As we can find in a line and planes defined in (18), geometric primitives are produced deductively. They have static structures in the vector space, and they never directly express geometric decision making in design. Geometric decision making in design is the very procedure to generate these geometric primitives. Therefore an object to express geometric decision(GeometricDecision) is introduced in GDL. That is, geometric decision making process in architectural design can be expressed by the graph of Geometric Decisions.

GeometricDecision does not indicate directly a particular geometric primitive, but addresses a deductive procedure of generating the primitive. It consists of 'parents, "context", "dependents', and "currentGeometry". The "parents" points out GeometricDecisions referred in the deductive procedure. The "context" stores procedures of geometric operations as data, and is expressed by so called "block" in Smalltalk language. The block is a particular object expressed by the statement in parentheses [, and not evaluated at the time of its declaration. A simple example of a block [x + y] indicates the procedure to add x and y, and the actual value depends on the values of x and y when it is evaluated. The "dependents" assigns a collection of GeometricDecisions generated from itself, and "currentGeometry' addresses the last evaluation of the "context".

Once receiving an "unsettled" message, the instance of GeometricPrimitive is promoted to an instance of GeometricDecision which stores the fact(context) of its production using characters of AI language in which we can handle procedures as data. In GDL , protocols which handle geometric Decisions also have the same usage to the protocols which operate GeometricPrimitives. For example, the geometric decision process which corresponds to (18) can be expressed as follows:

```
| decision1 decision2 decision3 |
decision1 := (Plane at: 0 @ 1 @ 0 normal: 1 @ 1 @ 1) unsettled.
decision2 := (Plane at: 1 @ 0 @ 0 normal: 2 @ -1 @ 1) unsettled.    (20)
decision3 := decision1 intersectionWith: decision2.
```

Here decision 1 is an instance of GeometricDecision whose parent is the current geometric origin(O @ 0 @ 0) and the context is a block [Plane at: 0 @ 1 @ 0 normal: 1 @ 1 @ 1]. Also decision2 is instantiated as a GeometricDecision whose context is a block [Plane at: 1 @ 0 @ 0 normal: 2 @ -1 @ 1]. While, decision3 stores the block of [decision 1 intersectionWith: decision2] which indicates the procedure to generate the intersection of decision1 and decision2. Its current geometry is also equal to (19):

$$(x) / 2 = (y + (1/2)) / 1 = (z - (3/2)) / -3 \qquad (21)$$

Translation or rotation of decision1 and decision2 will cause a new geometry in decision3, but it can be acquired to evaluate the stored context in case of a necessity. For example, the following operation:

```
decision1 moveBy: 4 @ 1 @ 3                                    (22)
```

triggers a change of current geometry in decision3 to:

$$(x) / 2 = (y - (7/2)) / 1 = (z - (11/2)) / -3 \qquad (23)$$

This change will be completed at the same time when the (22) message is executed, and dependent mechanism is introduced to achieve this propagation. In dependent

mechanism, every GeometricDecision which makes a certain change must propagate 'changed messages to its dependents. GeometricDecisions which receive this message evaluate their context and propagate "changed" messages to their dependents again. This mechanism will guarantee the geometric consistency.

When additional translation is performed to decision3:

```
decision3 moveBy: 3 @ 4 @ 5.                    (24)
```

context of decision3 will be represented as a nested block [{decision1 intersectionWith: decision2} moveBy: 3 @4 @ 5]. Since the nested block can be evaluated reflexively, decision history is also expressed by the nested structure of stacking geometric decisions on each geometric object.

6 Interaction to architectural model

OOAMS(Object-Oriented Architectural Modeler and Simulator) is a project with the purpose of developing future CAAD systems, and AKM is an architectural model employed as the core representation of OOAMS system, in which architectural elements are described from the stand point of cognitive science.

Current 3D-CAAD systems generally have no architectural model, and 3D data which basically includes no architectural attributes and performances are substituted for one. In their actual implementation, 2D data as a display model, i.e., bitmap or polygons, are produced from 3D data as a graphic model, i.e., surfaces or solids, by the geometric transformation in a rendering process. For example, their interaction may be expressed as follows:

rendering

Surface Model --> Display Model

Graphic Libraries, e.g., GL and PEX, can be a collection of representations and functions to achieve the above rendering operations.

In the actual implementation of OOAMS, representation of knowledge-based architectural model and description of architectural design process were mainly concerned, and a new mechanism was examined to simulate an architectural design convention in which architectural elements are usually drawn in different expressions in design specifications, i.e., drawings, at each state of its design process. The introduced approach was that individual architectural elements should have their intensive methods to generate 2D data for visualization respectively according to the designer's requests. These interaction can be expressed as follows;

production

Architectural Model --> Display Model

In AKM, architectural design description is considered a collection of architectural decision making. It can be represented with the collection of facts in which relations of architectural elements are described with formal predicates. Therefore predicates are the only authority to restrict spatial configuration of architectural elements, and each element will be an argument of design statements. GDL mentioned in previous chapters is intended for the knowledge-based geometric model which is able to contribute toward improving the degree of freedom in the description of architectural spatial configuration, and also be able to guarantee the geometric uniqueness of design descriptions. Their interaction could be expressed as follows;

regulation production

Geometry Model --> Architectural Model --> Display Model

Geometric decisions are separated from total descriptions of architectural elements, and they regulate the spatial positions of architectural elements. According to above interaction, the sample example of a procedure to define instances of Floor in am can be expressed as follows:

```
| decision1 decision2 floor1 floor2 |
decision1 := (Plane xyPlaneAt: 4800) unsettled.
decision2 := decision1 translatedBy: 0 @ 0 @ 3600.          (25)
floor1 := Floor supportedBy: decision1.
floor2 := Floor supportedBy: decision2.
```

Here, each floor is supported by an instance of GeometricDecision, and there is a restriction of "translatedBy: 0 0 @ 3600' between GeometricDecisions. Any translation of decision1 will not cause change in the distance between floor1 and floor2.

The orbit of an architectural column may be a line determined by two reference planes which indicate the orders of columns placed on a grid. Another example is the procedure to define a Colunm supported by the intersection(decision3) of two reference planes(decision l, decision2):

```
| decision1 decision2 decision3 column1 |
decision1 := (Plane at: 0 @ 0 @ 0 normal: 1 @ 0 @ 0) unsettled.
decision2 := (Plane at: 0 @ 0 @ 0 normal: 1 @ 1 @ 0) unsettled.    (26)
decision3 := decision1 intersectionWith: decision2.
column1 := Column supportedBy: decision3.
```

Here column1 is also supported by an instance of GeometricDecision. But in this case, every translation of decision1or decision2 causes change in the actual position of column1.

Even in the last interaction, every architectural object still includes the visualization procedures just as the transformation of coordinations. Therefore its rendering process will be performed locally by architectural objects. For collaborative design, a distributed concurrent system will be required. Procedures handled on a server should be as light as possible on a client-server computing environment. According to this interaction, it is difficult to prevent one from getting down the system performance in proportion to the number of access from clients, because a geometry model and an architectural model would be handled on a server. Rendering is a very heavy process which will display its result and is able to be handled on the client side. If an architectural model simply produces its graphic model, and abandons all rendering procedures, the interaction between each model will be expressed as follows;

regulation production rendering

Geometry Model --> Architectural Model --> Surface Model --> Display Model

Here the former two models shall be handled on the server side, and the latter two models shall be handled on the client side, then an ideal distributed design system can be reached.

7 Interface to input geometric decisions

Sample descriptions of geometric decisions quoted in this paper are written in explicit statements, but geometric decision making is usually implied in designing activity. How to input geometric decisions is another problem for teaching rules to the model for the actual use of GDL.

Currently, graphical user interface have been familiar in CAD/CG systems, and many graphic operations can be selected and executed by a certain pointing device. In these systems, geometric decisions are translated to equivalent commands for graphical operations. Therefore, individual command execution may be considered as the design decision. However, to store the sequence of command execution is one thing, and to input geometric decisions is another.

GDL is implemented as a simple class library, and there is no particular graphical user interface. It has been developed as a part of the OOAMS system, so all geometric operations are performed from OOAMS interface. OOAMS can be operated with a mouse and pop-up menus in a multi-window system, but there are no functions of graphical operations such as drawing lines, circles, or rectangles which CAD/CC systems ordinarily have.

Current version of OOAMS only deals with a rigid frame structure, and reference lines(planes) which constitutes a grid for columns should be produced first as GeometricDecisions. As a GeometricDecision is an object which stores geometric decision as a block, reference lines also reflect each geometric restriction.

In GDL, GeometricCoordination is prepared to coordinate GeometricDecisions. For example, the following statement:

```
| axis coordination |

axis := Line xAxis unsettled.                                    (27)
```

coordination := GeometricCoordination at: axis orthogonalized With: #(3600 7200 10800). instantiate a GeometricCoordination which coordinate three GeometricDecisions, where three geometric planes orthogonalized to the axis according to the progression are indicated.

In OOAMS, the above procedure is instructed from an usual dialog box, and a fundamental grid for architectural design is declared in the beginning of the architectural design process.

GeometricPrimitives are never handled directly. Every geometric operation is performed on an existing a GeometricDecision. Executing a geometric operation for a certain a GeometricPrimitive is equivalent to either instantiating another a GeometricPrimitive or updating its intensive values, whereas executing a geometric operation for a certain a GeometricDecision is equivalent to either instantiating another a GeometricDecision or encapsulating its last context. Since GeometricDecision has the same protocol of geometric operations as GeometricPrimitives, we can input geometric decisions as if we just handle GeometricPrimitives from the usual graphical user interface.

8 Conclusion

Throughout this paper, I have explained about principles and the implementation of GDL(Geometric Decision Library) which is necessary for future CAAD systems.

GDL objects constitute a geometric decisions network for architectural design. Reconsideration of any geometric decision for an architectural design extracts each restrictions, and can maintain total geometry in every state of the architectural design process. There still is the problem that the nested block is sometimes too complicated to reduce to an equivalent simple block after so many encapsulations are taken to the block. Another problem is how to delete a geometric decision which has already been obsoleted in a design description but still refered from other decisions.

This research was supported in part by a grant from the Ministory of Education of Japan. GDL is implemented on Smalltalk environment(Objectworks 4.1/VisualWorks 1.0), and is able to run on various platforms(UNIX, Windows, and Macintosh). Total source code size is about 150KB and will be available from our World-Wide Web server(http://gaudi.sk.tsukuba.ac.jp/).

9 References

Carrara,G., Kalay,Y.E., & Novembri,C. (1994). Knowledge-Based Computational Support for Architectural Design, Knowledge-Based Computer-Aided Architectural Design, Elsevier.

Code,P., & Yourdon,E., (1990). Object-Oriented Analysis, Yourdon Press.

Code,P., & Yourdon,E., (1991). Object-Oriented Design, Yourdon Press.

Cox,B.J., (1986). Object Oriented Programming: An Evolutionary Approach, Addison Wesley.

Coyne,R.D., Rosenman,M.A., Radford,A.D., Balachandran,M., & Gero,J.S. (1989). Knowledge-Based Design System, Addison Wesley.

Eastman,M.C. (1994). A Data Model for Design Knowledge, Knowledge-Based Computer-Aided Architectural Design, Elsevier.

Flemming,U., & Wyk,S.V.(ed.) (1993). CAAD Future 93, Proceedings of the Fifth International Conference on Computer-Aided Architectural Design Futures, NorthHolland.

Gero,J.S. (ed.) (1992). Artificial Intelligence in Design 92, Kluwer Academic Publishers.

Gero,J.S., & Sudweeks,F. (eds.) (1994). Artificial Intelligence in Design 94, Kluwer Academic Publishers.

Goldgerg,A., & Robson,D. (1985). Smalltalk-80 The Language and its Implementation, Addison Wesley.

Goldberg,A. (1985). Smalltalk-80 Interactive Programming Environment, Addison Wesley.

Rambaugh,J., et al. (1990). Object-Oriented Modeling and Design, Prentice-Hall.

Mitchell,W.J. (1990). The Logic of Architecture, MIT Press, Cambridge, Massachusetts.

Kalay,Y.E. (ed.) (1992). Evaluating and Predicting Design Performance, A WileyInterscience Publication.

Schmitt,G.N. (ed.) (1991). CAAD Future91, International Conference for Computer Aided Architectural Design, Education, Research, Application, Vieweg.

Straber,W., & Seidel,H.-P. (eds.) (1989). Theory and Practice of Geometric Modeling, Springer-Verlag.

Walfram,S. (1988). Mathematica -A System for Doing Mathematics by Computer-, Addison Wesley.

Watanabe,S., & Watanabe,H. (1986). Possibility of Artificial Intelligence and Knowledge Representation for Architectural Planning, Proceeding of 9th Symposium on Computer Technology of Information, System and Application, AIJ, pp.277-282, Tokyo Japan. (written in Japanese)

Watanabe,S., & Watanabe,H. (1989). Architectural Modeling with the Relationship of Concepts in the Framework for Representing Knowledge, Proceeding of 12th Symposium on Computer Technology of Information, System and Application, AIJ, pp.l93-198. Tokyo Japan. (written in Japanese)

Watanabe,S., & Watanabe,H. (1990). Extension of Architectural Knowledge Representation and Understanding Method of Meaning, Proceeding of 13th Symposium on Computer Technology of Information, System and Application, AIJ, pp.139-144, Tokyo Japan. (written in Japanese)

Watanabe,S., & Watanabe,H. (1991). Architectural Concept Modeling in the Framework for Representing Knowledge, Proceedings of the 4th International conference on Computing in Civil and Building Engineering, pp.18S-l92, Tokyo, Japan.

Watanabe,S. (1992). Architectural Design Environment in 2010, Journal of Architecture and Technology, Delphi Research. (written in Japanese)

Watanabe,S. (1994). Knowledge Integration for Architectural Design, KnowledgeBased Computer-Aided Architectural Design, Elsevier, pp.123-l46

Yoshikawa,H. (1987). General Design Theory, Journal of Artificial Intelligence, Vol. 2, No. 3, pp.12-23. (written in Japanese)

Yoshikawa,H., & Tomiyama,T. (eds.) (1990). Intelligent CAD, Asakura shoten. (written in Japanese)