DesignBUF: Exploring and Extending 2D Boolean Set Operations with Multiple Modes in the Early Design Phase

Jin Won Choi, Do-Young Kwon[1] and Hyun-Soo Lee[2]
Yonsei University, [1]Ajou University,[2]Yonsei University

Key words: Design Buffer, Extended Boolean Set Operations, Structured Floor Plan.

Abstract: Boolean set operations have been a powerful design function set for any CAD systems including 2D and 3D domains. Their capacity to provide even more powerful design tools have not, however, been fully explored in the 2D system. The purpose of this study is to further explore 2D Boolean set operations with multiple modes, which include a pick mode, a wait mode, a drag-and-drop mode, and a draw-and-action mode. We develop a prototype design tool, called DesignBUF. It introduces a new concept of "design object buffer," an intermediate design zone in which a designer freely sketches his/her design with design objects in a brainstorming fashion since valuable design ideas are ephemeral? and the designer needs to generate design schemes rapidly before the ideas disappear or are forgotten. After finishing such fast brainstorming processes, especially in the early design phase, the designer gets a stable and refined form of a floor plan, which in turn becomes a well structured form to maintain building and design information systematically. Therefore, the designer keeps switching back and forth between the "design object buffer" and structured floor plans. We believe that this dual working memory will not only increase system flexibility, but also reduce computation with unnecessarily complex design objects. This study also develops a robust algorithm to transform the intermediate design objects into a well-structured floor plan. In fact, the algorithm is also used for the extended Boolean set operations described above. A structured floor plan can also be transformed into non-structured forms. Research issues for future development are also identified at the end of the paper.

1. THE CREATIVE DESIGN PROCESS

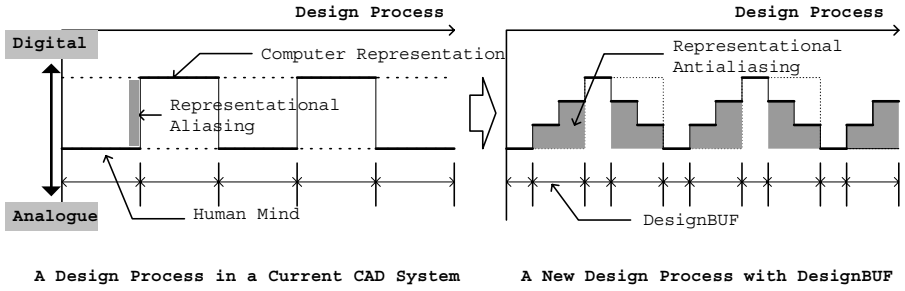


Figure 1. DesignBUF for Representational Antialiasing

A well-known notion is that there are differences between the human mind and computer representations. The differences create a certain gap between how designers think and how computers work. Interfacing these two worlds is critical in developing any design systems. However, most current CAD systems obviously present such aliasing problems (Figure 1). Researchers have attempted various antialiasing strategies, one of which is to allow design flexibility in the early design process. Assume that any architectural design process deals with two different design objects: graphic and architectural. Graphic entities are more appropriate for the early design stages due to their semantically-free and sketch-like attributes, while architectural entities are more adequate for the design development stages due to their semantically-specific and component-based attributes. These two entities often interact and switch back and forth as the design process proceeds. A problem of current CAD systems is the lack of such interactions. It might be important to provide an intermediate design zone wherein two types of entities coexist and can be freely moved back and forth by designers. We call the zone a design buffer or DesignBUF.

Table 1 Graphic Entity versus Architectural Entity

Graphic Entity	Architectural Entity
2D Graphic Systems	Architectural CAD Systems
Semantically-free	Semantically-specific
Not object-oriented	Object-oriented
Suitable for early design stages	Suitable for design development stages
Does not have to be structured	Need to be structured
Can support many CAD theories such as shape grammars, shape emergence, space syntax, etc.	Difficult to support them

DesignBUF also supports a top-down approach, an important design strategy in architectural design. It not only allows designers to develop a design from graphic elements to architectural elements, but also helps designers to develop mass models from? architectural models in a rapid and automated manner.

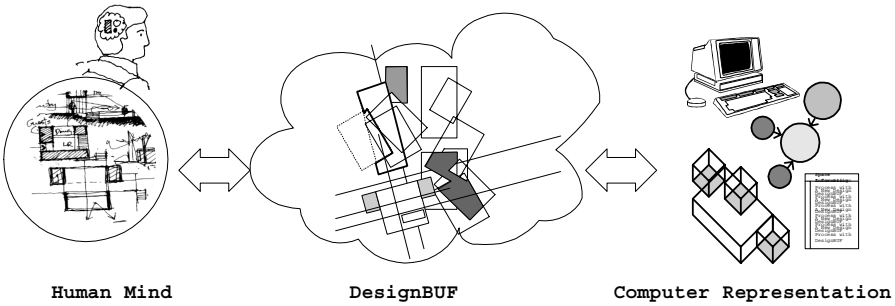


Figure 2. DesignBUF between Human Mind and Computer Representations

2. RELATED WORKS

This study mainly relates to two critical research issues: 1) The form generation process based on mathematical rules and enhanced user interactions, and 2) Space-based representations.

2.1 The Form Generation Process

Shape emergence is a well-known research issue in the form generation process. Tan (1990) discusses the centrality of emergent forms in the design process and proposes a data structure based on construction lines and ordered lists which enables shapes as a collection of lines and arcs to be efficiently encoded. Shape grammars obviously relate to emergence issues in terms of the continuity of computations (Stiny, 1993).

ReDraw (Kolarevic, 1994) is a computer-based graphic environment for design conceptualisation, especially for shape delineation and dynamic drawing manipulation, and is based on construction lines and their geometric relations.

Chase (1996) describes a way of using shape algebras and emergence instead of more traditional representations of geometric objects. His method attempts to describe designs combining the paradigms of shape algebras and predicate logic representations.

He (1999) also investigates better user interaction modes in the grammar-based design process. He proposes possible modes of user interaction and control within grammar-based design systems.

2.2 Space-based Representations

There has been some research on developing CAD systems that can build structured floor plans (Yessios, 1986a & 1986b; Choi, 1997; Kalay et al, 1998). In such systems, a floor plan is structurally well defined by having some hierarchical components. Since space and form are the two main aspects to define a building, they should be represented at the same time within a system.

A space-based representation is also examined in Mahdavi's recent research (1999). Trying to integrate detailed simulation methods and CAD systems, he recognized the importance of spatial information. He observed that detailed thermal simulation methods require the definition of spaces and zones, and not just bounding surfaces.

Carrara's research (1994) focuses on using spatial information in the very first stage of the design process. The concepts of Space Units (SU) and Building Units (BU) that the researchers employed allowed the system to represent the defined SUs as bubbles, highlighting the adjacencies and the defined paths.

An interactive graphic approach (Liggett, 1992) has been attempted to solve spatial allocation problems in facility layout. The study introduced the concepts of 'stack plan', 'zone plan', 'block plan' and '1to1 plan'. To generate an ideal block plan, a designer employed a process of generation, adding/modifying criteria, and appropriate trade-offs in an iterative, interactive fashion.

3. EXTENDING BOOLEAN SET OPERATIONS

3.1 Boolean Set Operations

Boolean set operations have been a powerful design function set for any CAD systems including 2D and 3D domains. They are particularly useful operations in any solid modelling system. Some general types of Boolean set operations include union, intersection, difference, and split. The general algorithm of each operation defines where to start tracing and how to trace line edges. For example, the algorithm of the union operation can be described as follows:

1. *Start from a vertex on the first object where the starting vertex is located at the outside of the intersected area.*
2. *Trace line edges in the positive direction until an intersection vertex encounter.*
3. *Jump to the other object and trace line edges in the positive direction until an intersection vertex encounters.*
4. *Repeat 2 and 3, and finish the process if it reaches the starting vertex again.*

While the operations are simple and clear to use in any form generation stages, they have not yet been fully explored to provide even more powerful design tools, especially in 2D architectural CAD systems. One of the most critical limits is that the operations deal with only two objects at a time. This could be a problem when a designer deals with many design objects, wants to figure out the relationships among the objects, and tries to find derivative forms based on the objects. The interaction mode is also limited even though some different mode types can be considered.

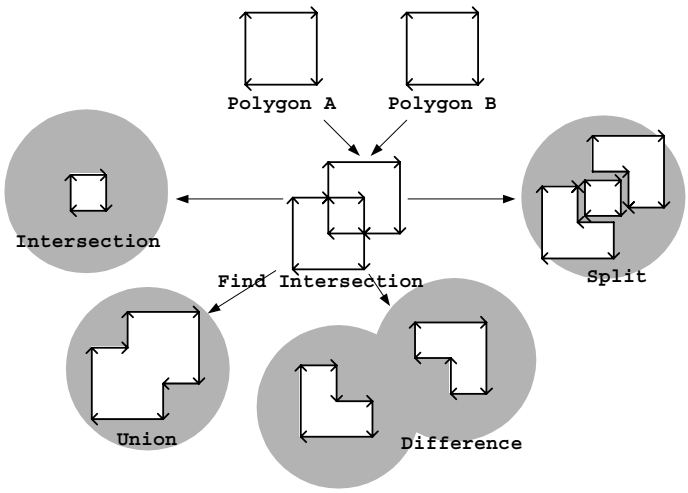


Figure 3. Boolean Set Operations

3.2 An Extension of Boolean Set Operations

As the term "Boolean" denotes, the Boolean set operations deal with only two objects. While the concept is simple and clear to understand, it often limits designers to develop various forms in a creative fashion.

The extended version of the Boolean set operations in this study can thus deal with multiple objects simultaneously. Along with the original function

set including union, intersection, difference, and split, a new function type, called 'offset, ' adds its power. The offset function converts? a line or a polygon to a polygon object with a width. The function sets we developed also work with two types of objects, void and solid. If we consider the solid type as a normal object, the void object is a new type where the line edge information is important and there is no spatial information related to the object. The sets also work with multiple modes in terms of user interactions.

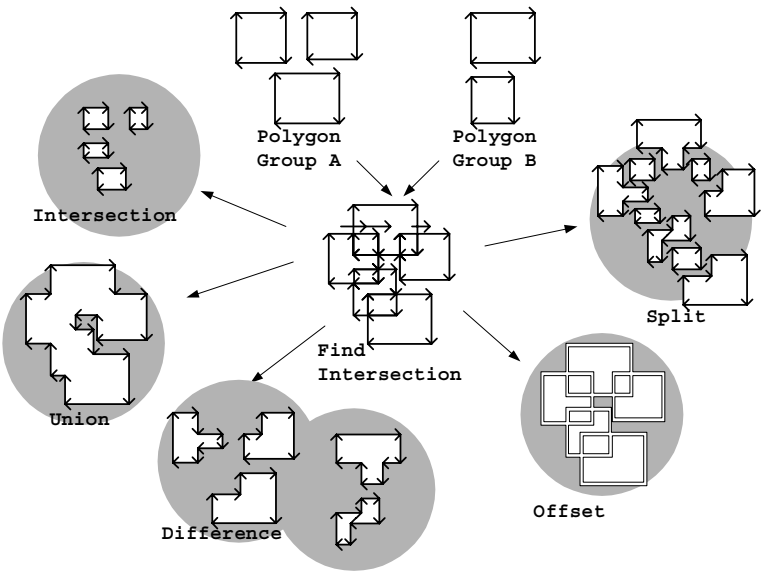


Figure 4. Extended Boolean Set Operations

3.3 Void and Solid Boolean Set Operations

If we assume that current 2D Boolean set operations only deal with solid objects filled up with something, we can consider void objects where nothing is inside. A solid object has a selectable area, filled with a user-defined colour. On the contrary, a void object has a transparent area. Operations work with objects of the same type. That means solid objects only work with solids. The following figure shows how two types of operations result in different outputs.

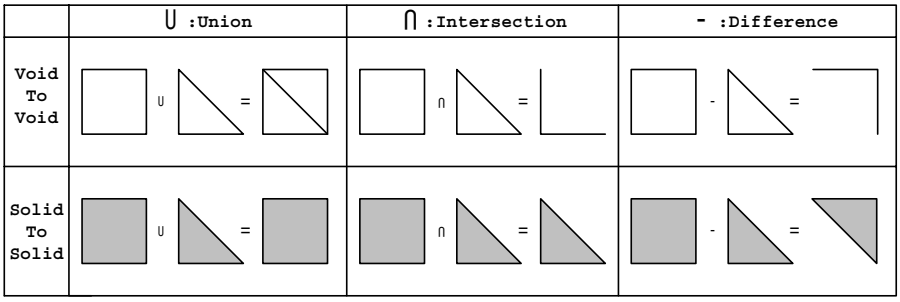


Figure 5. Types of Solid and Void Boolean Set Operations

The following figure presents a process using extended Boolean set operations. The process starts with creating polygons, followed by structuring polygons. The next step is to specify spatial information. The outputs can be represented in different ways: void/solid objects, a space diagram, 3D models, a floor plan, etc.

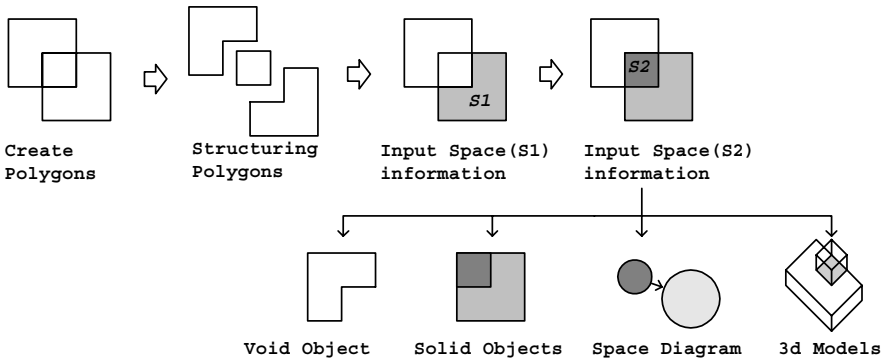


Figure 6. A Process Using Solid and Void Boolean Set Operations

3.4 Multiple Modes

Four types of multiple modes are based on the action point of the time line in the early design phase. A pick mode, a first among four modes of the extended Boolean set operations, is a normal mode shown in any other CAD system. The designer needs to pick two design objects to conduct a Boolean operation. In a wait mode, the system waits until the designer asks to conduct a Boolean operation after finishing a set of editing operations, such as transformation and rotation. Further, a drag-and-drop mode allows the designer to simply drag and drop design objects onto other objects in order to activate a given Boolean operation. This mode works well with the

symbol and block concepts. Using symbol/block databases, one can design many alternate design solutions. And finally, in a draw-and-action mode the designer draws an object on existing objects and the system automatically processes a Boolean operation. The designer can work in any mode among those four during any moment of the design process.

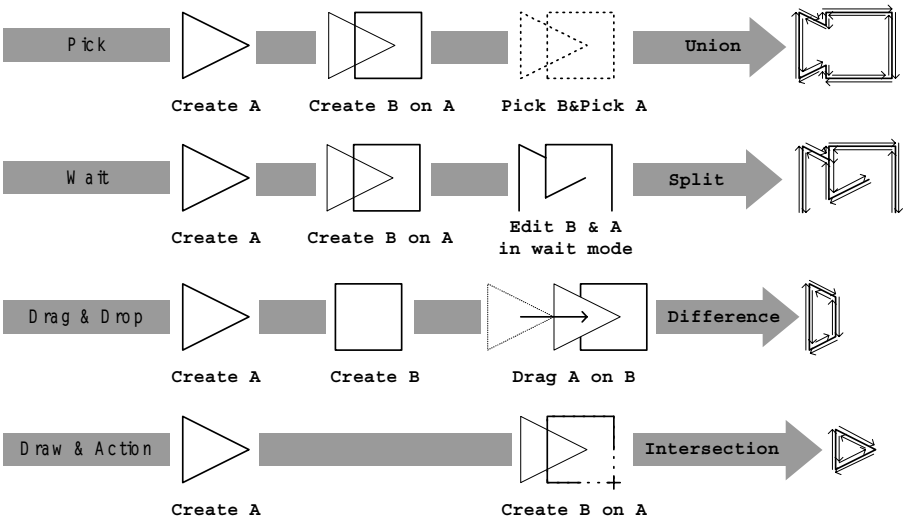


Figure 7. Extended Boolean Set Operations with Multiple Modes

4. THE BASIC ALGORITHM FOR EXTENDED BOOLEAN SET OPERATIONS

4.1 Basic Strategies in The Process

The following figure presents how original objects are divided into multiple spatial polygons that have relationships with each other. Overlapped objects (A and B) generate intersected areas(C). They also construct a set of spaces including an outside space representing a boundary of the total objects. We have developed? an algorithm that automatically structures and represents the spaces included.

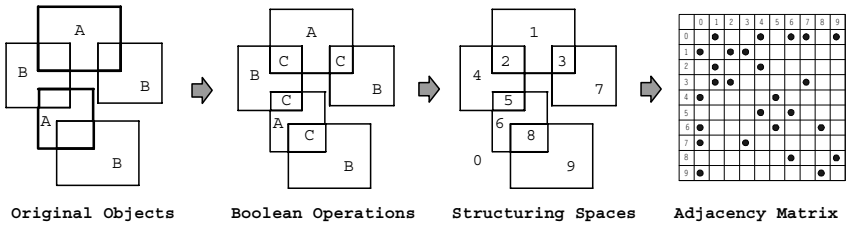


Figure 8. The Process of Extended Boolean Set Operations

The algorithm for extended Boolean set operations focuses on dealing with multiple objects at the same time and structuring spatial information automatically. The basic strategies are, therefore, as follows:

1. The input geometry is composed of a set of several input lines that may or may not have intersections with the existing geometries and even self-intersections with the input lines themselves. In Figure 10, input points 1 to 7 are a set of input geometry.
2. A set of input geometry is instantly processed to maintain a structured form.
3. A set of input geometry is divided into several units, taking into account intersections with existing geometries and self-intersections. A geometrical unit, therefore, does not have any intersections. Figure 10 shows five geometrical units divided.
4. There are four types of intersections with existing geometries: T, Y, I and N.
5. A special consideration is required in case both ending points (a starting point and a target point) of a geometrical unit are located on an identical existing geometry.
6. A special consideration is also needed with inputs paralleled and attached to existing geometries.
7. Existing rings through which ending points pass should be recognized. A special consideration is required when the existing rings differ from each other.
8. Vectors of geometrical surfaces should be rerouted in a defined fashion based on the intersection type.
   Following is a description of the process in detail.

4.2 The Break-Down of The Input Geometry

Input geometries are a sequence of input line*s* given by the designer and allows self-intersections with the input lines themselves. For the simplicity

of the creation algorithm, these input lines need to be broken down into a set of geometrical units with no intersections at all.

The example in Figure 10 presents complex input geometry on existing geometries. The breakdown process divides the input geometry into five units. The following creation process is repeated five times to finish the whole process.

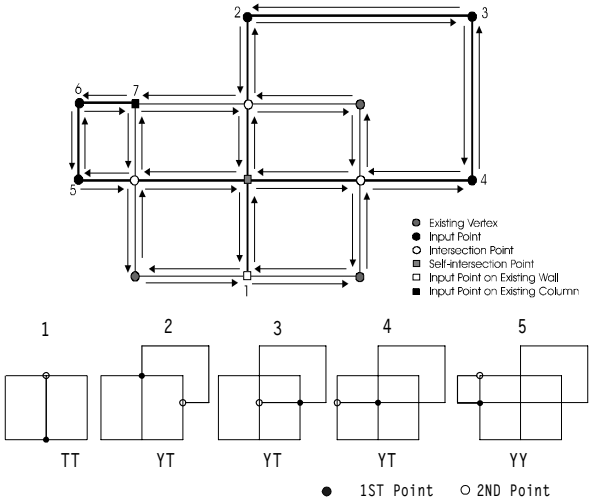


Figure 9. The breakdown process of the input geometry

4.3 Intersection Types in a Geometry

Each point in a geometrical unit, having two end points, a starting point and a target point, has one of four intersection types such as T, Y, I or N. Type T occurs when the input point is located on an existing geometry, while the input point of Type Y is on an existing column where two existing geometries are crossing one another. In Type I, the input point is placed on the end of an existing geometry where two geometries are not crossing. The input point on Type N is freely located on a ring detached from any geometry.

Table 2 Intersection Types in a geometry unit

2ND / 1ST	T	Y	I	N
T				
Y				
I				
N				

4.4 Rerouting Geometry Surface Vectors

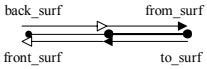
Once intersection types are recognized, vectors of geometrical surfaces connected to the input points are rerouted to maintain a correct structure. The rerouting process occurs in three intersection types excluding Type N.

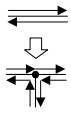
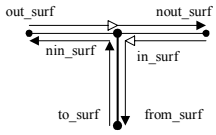
First, a relatively simple rerouting process occurs for Type I. When from_surf and to_surf are geometry surfaces belonging to the input line, back_surf is connected to from_surf and to_surf to from_surf.

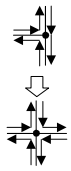
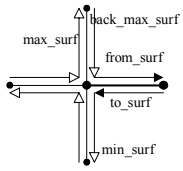
Next, the process for Type T makes one wall and two geometry surfaces, nout_surf and nin_surf.

Lastly, for Type Y it is important to find two vectors, one closest to the input line in a clockwise direction and the other in a counter-clockwise direction. To do this, it is necessary to calculate angles between each geometry and the input line after collecting all surface vectors connected to the vertex.

Table 3 Rerouting wall surface vectors for intersection types

Type	Scheme	Rerouting Process	Pseudo-code
Type I			back_surf->next = from_surf; to_surf->next = front_surf;

Type T			out_surf->next = nout_surf;
			in_surf->next = from_surf;
			to_surf->next = nin_surf;

Type Y			to_surf->next = min_surf; back_max_surf->next = from_surf;

5. CONCLUSIONS AND FUTURE RESEARCH

5.1 Conclusions

Based on the notion of the gap between the way designers work and the way computers represent, this study explores a way of increasing design flexibility. The method we have developed? is based on an extended set of 2D Boolean set operations which, we believe, allow designers to start with some graphical design elements, develop emergent forms using such operations, and end with a well developed model with architectural elements generated from the semantically-free forms previously generated?. The method introduces void objects opposite to existing solid objects. Considering these two types of objects instead of only one would give more chances for designers to think and develop forms in different ways. Multiple user interaction modes also add flexibility since designers can delay or process their operational decisions at any time in the design process. As an end result, this set of operations would add diversity in terms of functionality if applied in any architectural CAD system.

In fact, this not only increases design flexibility, but also improves design productivity since the method integrates 2D and 3D design environments. That is, it automatically generates 3D architectural model*s* based on 2D floor plan data.

This method would allow designers to evaluate building performance from various perspectives. It builds a structurally well-defined floor plan that has information about spaces within the floor plan and their relational

information. Based on such information, various building performance evaluation is possible when individual evaluation modules are developed.

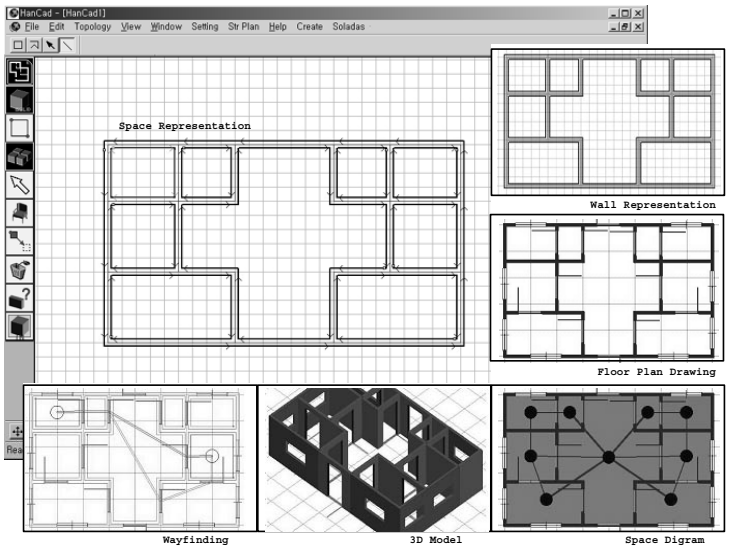


Figure 10. The Use of DesignBUF

5.2 Future Research

The system, currently under development, has two different versions: a standalone version written by C++ and a Java applet version. The Java applet also includes other research issues as a tool for a collaborative design.

Further research issues include developing a mechanism to efficiently record design changes as a design evolves and move from one design to the other with a set of recorded designs.

The approach applied in this study, mainly based on a space-based floor plan representation of buildings, has extensive potential application areas. Although some of them are partly implemented and tested in the DesignBUF environment, many others still need to be explored. For example, based on spatial information, DesignBUF can automatically construct floor plans with 3D building information that in turn generate a 3D solid model. Beyond that, we are exploring a way of managing a floor plan and a 3D model simultaneously. When a change is caused in a floor plan, another corresponding operation is transparently activated to adjust the 3D model.

6. REFERENCES

Branko Kolarevic: 1994, Lines, Relations, Drawing and Design, *Reconnecting,1994 ACADIA Proceedings*, p.51-61

Carrara, G. & Kalay Y.: 1994, Knowledge-based computational support for architectural design, *Automation in Construction*, 3, p.157-175

Choi, J.W.: 1997, A development of an intelligent CAD engine to support architectural design collaboration, *Korea CAD/CAM Journal*, p.53-59.

Eastman, C.M. et al.: 1991, *A Data Model for Design Databases*, *in* J.S. Gero (ed), Artificial Intelligence in Design '91, Butterworth- Heinemann, Oxford.

Eastman, C.M.: 1991, A Data Model Analysis of Modularity and Extensibility in Building Database, *Environment and Building*, 27(2), pp.135-148.

Eastman, C.M.: 1994, A Data Model for Design Knowledge*, Automation in Construction*, 3, pp.135-147.

George Stiny: 1993, Emergence and Continuity in Shape Grammars, *CAAD Futures ' 93*, p.37-54

Kalay, Y, Khemlani, L, & Choi, J.W.: 1998, An Integrated Model to Support Distributed Collaborative Design of Buildings, *Automation in Construction, 7*, pp.177-188.

Kim, U.: 1992, An Object-based Modeling System Coupled with Design Information System, *6th International Conference on Systems Research*, Baden Baden, Germany.

Kim, U.: 1995, *An Object-based Building Product Model : From Modeling to Documentation*, International Council for Building Research Studies and Documentation, W78 TG10 Workshop, Stanford, U.S.A.

Liggett, R.S.: 1992, *Designer-automated algorithm partnership: an interactive graphic approach to facility layout*, *in* Kalay, Y. (ed), Principles of Computer-Aided Design: Evaluating and Predicting Design Performance, John Wiley & Sons, Inc.

Mahdavi, A.: 1999, A comprehensive and computational environment for performance based reasoning in building design and evaluation, *Automation in Construction*, 8, pp.427-435

Maher, M.L. & Simoff, S.J.: 1997, Formalising building requirements using Activity/ Space, *Automation in Construction*, 6, pp.77-95

Milton Tan: 1990, Closing in an Open Problem – reasons nad a strategy to encode emergent subshapes, *1990 ACADIA Proceedings*, p.5-19

O'Neill M.J.: 1992, *Neural network simulation as a computer-aided design tool for predicting wayfinding performance*, *in* Kalay, Y. (ed), Principles of Computer-Aided Design: Evaluating and Predicting Design Performance, John Wiley & Sons, Inc.

Scott C. Chase: 1996, Design Modeling With Shape Algebras and Formal Logic, *Design Computation, 1996 ACADIA Proceedings*, p.99-113

Scott C. Chase: 1999, Grammar Based Design: Issues for User Interaction Models, *Media and Design Process, 1999 ACADIA Proceedings,* p.198-210

Steadman, P.: 1993, A database of the non -domestic building stock of Britain, *Automation in Construction*, 2 (1) pp.31-45.

Yessios, C.I.: 1986a, *The Computability of Void Architectural Modeling*, The Computability of Design, Proceedings of Symposium on Computer-Aided Design at Suny Buffalo, New York.

Yessios, C.I.: 1986b, *Architectural Modeling, Architecture 844 Lecture Notes I*, Graduate Program in Computer-Aided Architectural Design, Department of Architecture, The Ohio State University.