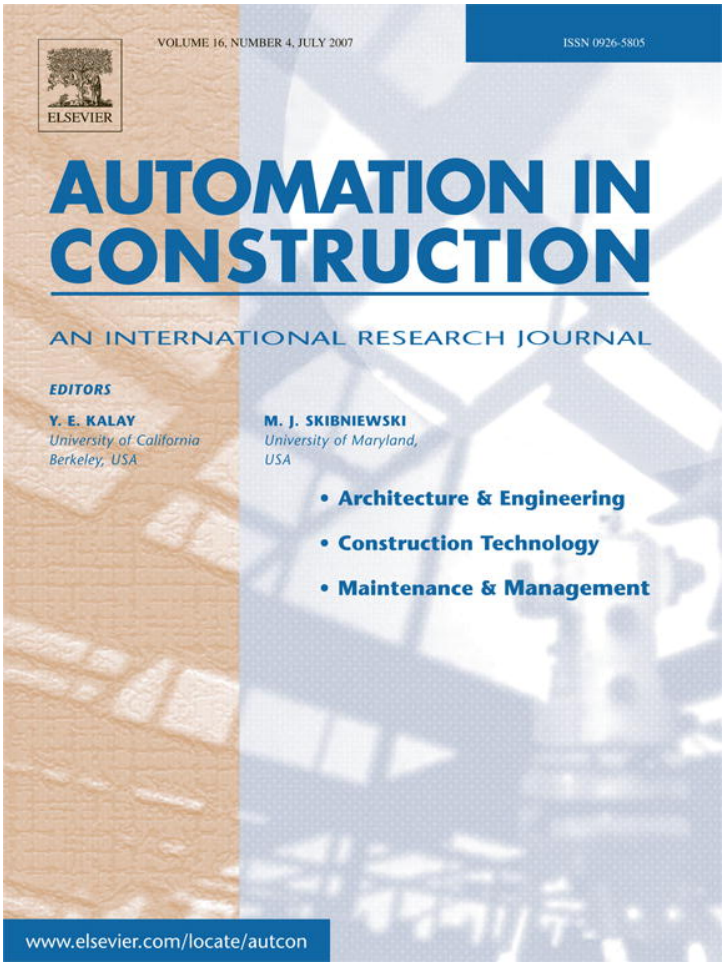


ELSEVIER

Automation in Construction 16 (2007) 449–459

AUTOMATION IN CONSTRUCTION

www.elsevier.com/locate/autcon

Real-time management of spatial information of design: A space-based floor plan representation of buildings

Jin Won Choi [a,*], Doo Young Kwon [b], Jie Eun Hwang [c], Jumphon Lertlakkhanakul [a]

[a] *Department of Housing and Interior Design, Yonsei University, 134 Shinchon-Dong, Seodaemoon-Ku, Seoul 120–749, South Korea*
[b] *Computer Graphics Laboratory, ETH Zurich, Switzerland*
[c] *Graduate School of Design, Havard University, Cambridge, MA, USA*

Accepted 14 August 2006

Abstract

The current trend towards globalization forces the AEC community to change its practical approach from a non-systematic, labor-intensive to a systematic one. Based on the paradigm shifted from 'drawing' to 'constructing' in architectural design practice, this paper focuses on developing a new type of CAD system that automatically constructs and manages well-structured floor plans with minimum geometrical input from the designer. The paper also took advantage of the building data models developed from the prior research. The model includes hierarchical building components such as 'building', 'plan', 'space', 'ring', 'wall skeleton', 'surface', 'column', etc. The creation algorithm developed assures a semantically rich and structurally correct floor plan at any point in the design process. In particular, the floor plan constructed through the design process contains spatial information as well as other design information about the building components. Thus, the system effectively manages spatial design information in the real-time basis. Since the system implemented on the basis of the algorithm is the very first step toward a complete intelligent CAD system, research and development issues to be considered next are identified at the end of the paper.

Keywords: Spatial information; Floor plan representation; Building data model; Component-based CAD

1. Introduction

The current trend towards globalization forces the AEC community to change its practical approach from a non-systematic, labor-intensive to a systematic one. The need for the paradigm shifted from 'drawing' to systematically 'constructing' for design practices is also recognized. However, the dominant usage in the building industry is still based on using CAD as a graphics editor. [3] Consequently, most CAD systems only support drawing works. They only manage graphical information, and are weak to consistently manage design information being generated during the design process. Due to this weakness, drawings are often not consistent and one change in a drawing consequently causes too many changes in other drawings.

To solve these problems, current trends in the CAD software industry show a main stream approach based on an object-oriented paradigm that integrates 2D and 3D data and representations. Some commercial software packages, such as ArchiCAD, All Plan, AutoCAD ADT, are based on the object-based paradigm integrating 2D and 3D design information. Research efforts [1,4,8,11,12] in the academia have also showed great interest in the issues that include how to systematically manage spatial information and how to manage and deliver design information by standardized formats such as the IFC (Industry Foundation Classes) model.

The goal of this study is to develop a computational mechanism which defines design objects as building components such as walls, slabs, columns and openings based on the object-oriented paradigm, and constructs a building consistently and systematically with the design objects. Special emphasis is placed on developing a computational method to manage spatial design information instantly and consistently that has been long time ignored in the conventional CAD systems. Here, a space means a room enclosed with walls, such as a living room

* Corresponding author. Tel.: +82 2 2123 3137; fax: +82 2 313 3139.
E-mail addresses: jchoi@yonsei.ac.kr (J.W. Choi), dkwon@inf.ethz.ch (D.Y. Kwon), jhwang@gsd.harvard.edu (J.E. Hwang), jumphon@yonsei.ac.kr (J. Lertlakkhanakul).

0926-5805/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.autcon.2006.08.003

or a dining room, or outdoor space defined by outside wall envelope.

The researchers observed two things: one is that conventional architectural design focused on the floor plan design; and the other is the notion of architect as composer. Louis Kahn, a great modern architect, mentioned in a lecture in 1966 that an architect should be a composer who composes with building elements as design entities. Based on the observations, this study employs a design method to construct a floor plan with design objects that automatically define a complete three-dimensional building.

This study is the first step toward a complete intelligent CAD system and focuses on how to generate a simple well structured floor plan. Therefore, the scope of the research is limited to the construction of a single-story building. More advanced features such as editing a floor plan fluently and evaluating it with various design criteria could be the next research topics.

2. Related works

There has been some research on developing CAD systems that can build structured floor plans [2,4,11,12]. 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. To be more effective it could take an object-oriented approach where each building component is an object from the view of the object-oriented paradigm. That is, the object has its own data and methods of how to behave in certain situations.

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. Almost all currently available commercial CAD systems rely on building representations that do not include spaces.

Carrara's research (1994) focuses on using spatial information on 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 [7] was attempted to solve spatial allocation problems in facility layout. The study introduced the concepts of 'stack plan', 'zone plan', 'block plan', and 'one to one plan'. To generate a best block plan a designer employed a process of generation, adding/modifying criteria, and appropriate trade-offs in an iterative, interactive fashion.

3. The concept of the 'structured floor plan'

The core of the study demonstrates how to construct a well-defined, well-structured floor plan (Fig. 1). In this paper we refer '*structured floor plan*' [2] to a floor plan composed by the designer in which its components are well structured and thus effectively express its architecturally meaningful structure. Such a plan has the following characteristics.

1) Object-oriented: The designer constructs a floor plan with predefined design objects such as walls, columns, slabs, and openings.
2) Definition of relationships between building components: It defines relationships between building components as design objects as well as their geometrical locations.
3) Management of spatial information: Spatial information as well as formal information is important in architectural design. Thus, it automatically manages spaces enclosed with wall components.

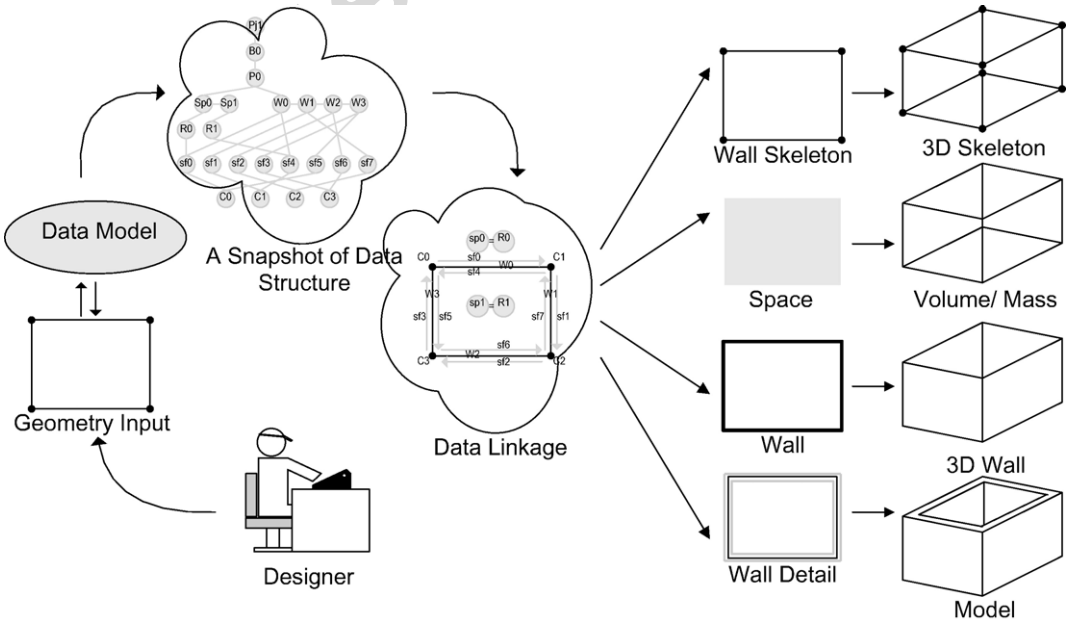


Fig. 1. The concept and principles of the 'structured floor plan'.

4) Management of spatial relationships: It shows the relationships among spaces in a floor plan, such as adjacency and connectivity with windows.
5) Instant and consistent management of design information: It manages design information in the automatic, instant, consistent manner at the moment the designer creates and edits design objects.
6) Levels of detail: When using computer-aided design tools, a designer is incessantly overloaded by a variety of aspects such as design alternatives, excessive details and organizational procedures. Proper level of detail is provided so that the designer is not overwhelmed by the details of the represented object. The system supports views where the abstraction in effect is the increase or decrease level of detail.
7) Multiple representations: The system handles multiple object views in a consistent and convenient fashion. A designer is able to manipulate an identical design object while he sees it from different viewpoints. The designer can see the object graphically, as a diagrammatic graph, or as a table of attributes, depending on his/her preference. Different types of representation can be mixed when an object is seen as an aggregate of component objects. On the other hand, a variety of representations can be presented in parallel in order to enhance the information delivery.
8) Partial modification: A specific component of a design object can be easily modified. When a partial modification takes place, the effect of change will be propagated to other objects when needed.
9) Automatic generation of a 3D model: It can automatically generate a 3D building model since it also includes 3D building information.

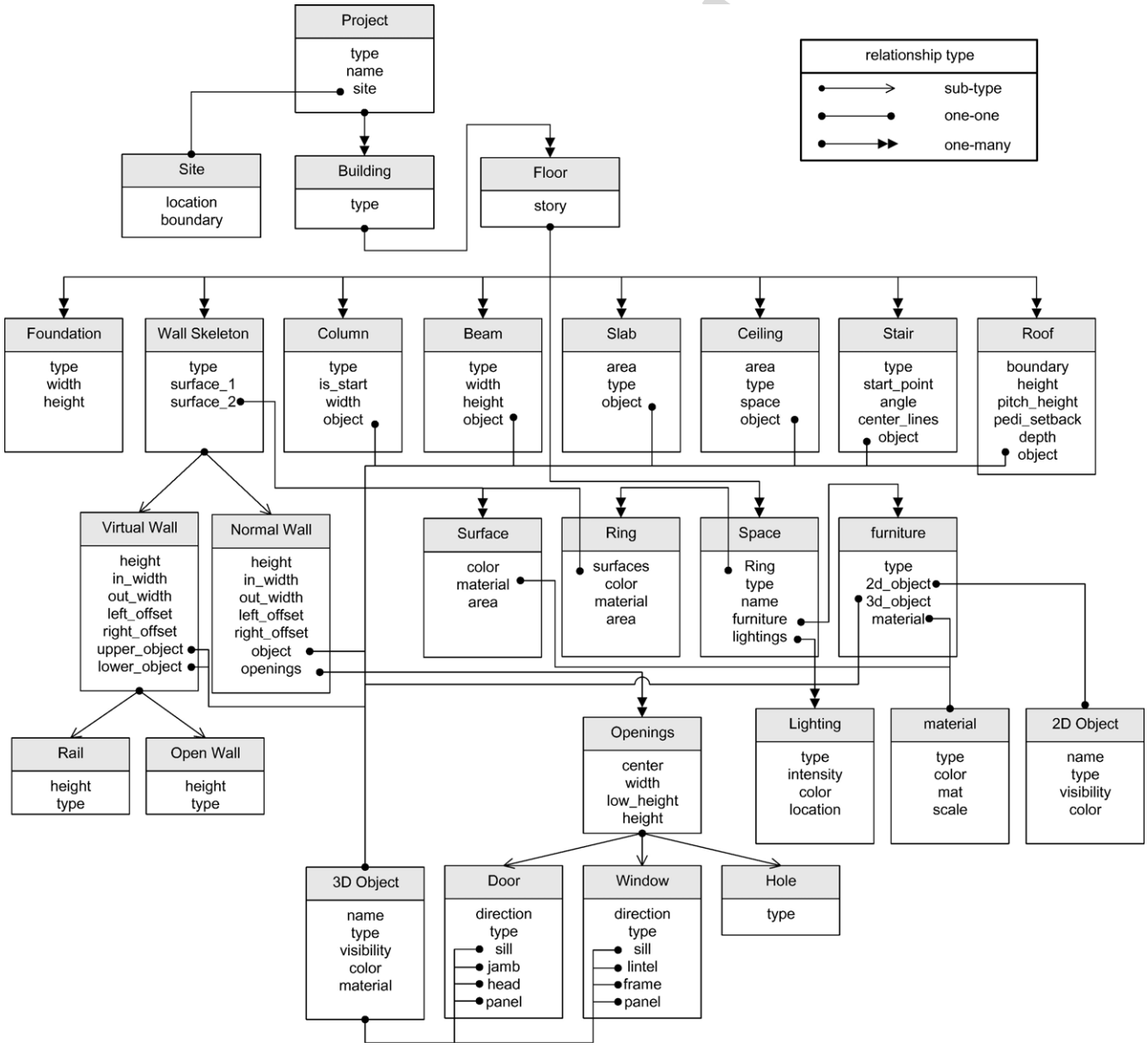


Fig. 2. The building data model for the structured floor plan.

10) Building data model-based: Its building representation is based on an intelligent space-based data model specifically designed for the study.

4. The building data model for the structured floor plan

4.1. The building data model

Fig. 2 illustrates a building data model developed through the study. Its structure is hierarchical and object-oriented, and it plays an important role in containing design information for each design project during the design process.

First of all, a project is the root of all class and can be composed of more than one building located on the same site. Each building contains several floors (plans) consisting of sub-building components including foundation, wall skeleton, column, beam, slab, ceiling, roof, stair and space. Among those components, wall and space are the most considerable since they embody the majority of building components. Each wall contains inside surface and outside surface. There are two kinds of wall. Normal wall is an ordinary wall which can contain openings. Such openings refer to doors, windows and holes. On the other hand, virtual wall means rail and open wall. Open wall is an assumed wall used to divide a room in to several spaces. A space generally has a single ring, but could have several rings as it contains holes inside. A ring is a closed polygon enclosed by a group of wall surfaces. Furniture, lighting and other appliances are located in a space. Each object has its own properties including materials. Fig. 3 illustrates basic components of the structured floor plan.

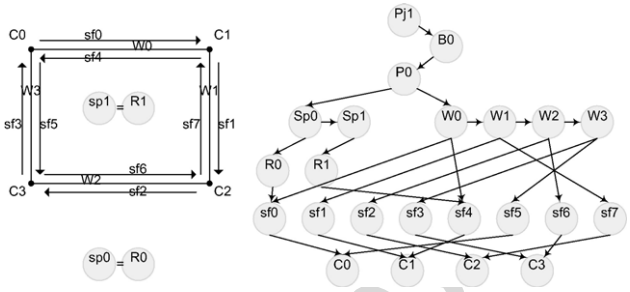


Fig. 4. A simple example of the structured floor plan.

4.2. Simple examples of the structured floor plan

As shown in Fig. 4, as soon as a designer draws a rectangular shape, the system constructs a room with four walls based on the building data model. By default, the room is set to be a part of 'project_1' (Pj1) located in 'building_0' (b0) on 'ground floor' (p0). This example shows that a simple single-story building is constructed internally as well as externally. The diagram illustrates two spaces, the first one for outside space and the other for inside space composed of a set of wall surfaces.

Once the model has been edited, the building data structure is also modified right after the designer inputs data. Fig. 5 shows the floor-plan and building data structure revised from the previous model shown in Fig. 4. As a wall (W4) has been inserted by the designer, new walls (W5,6) and surfaced (sf8–sf13) are autonomously created. The new space (Sp2) and ring (R2) are also inserted by the system. Finally, openings (d1-2, wn1-2) are manually added by the designer. The spatial

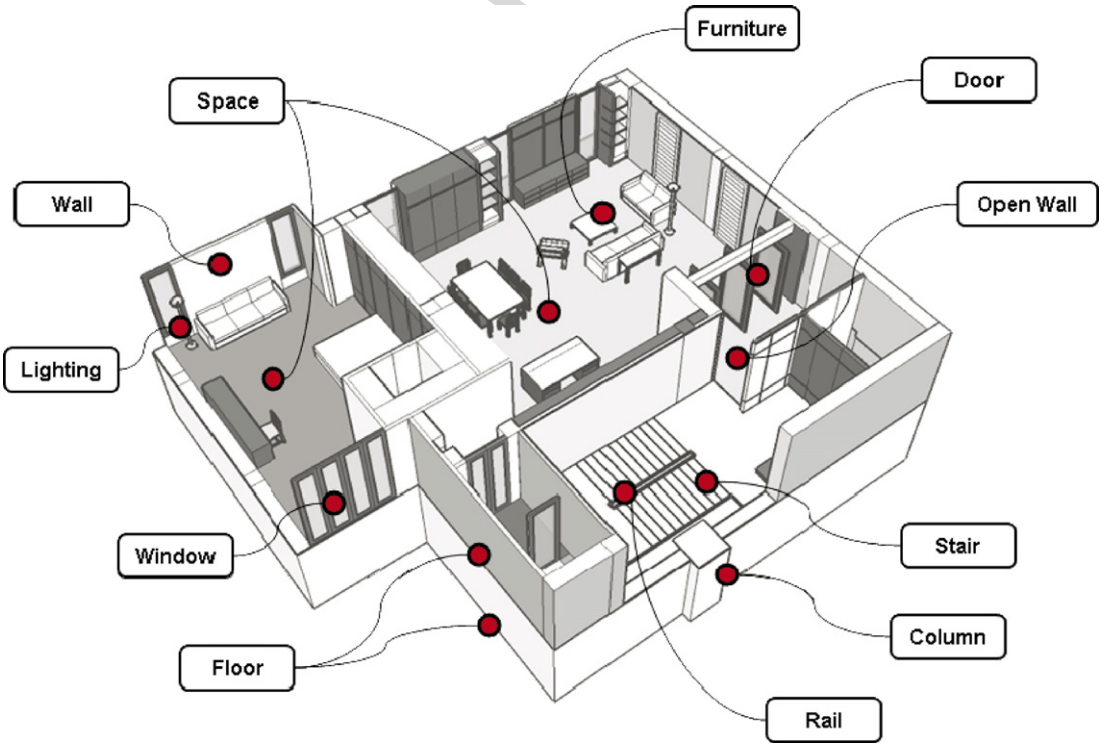


Fig. 3. The basic components of the structured floor plan.

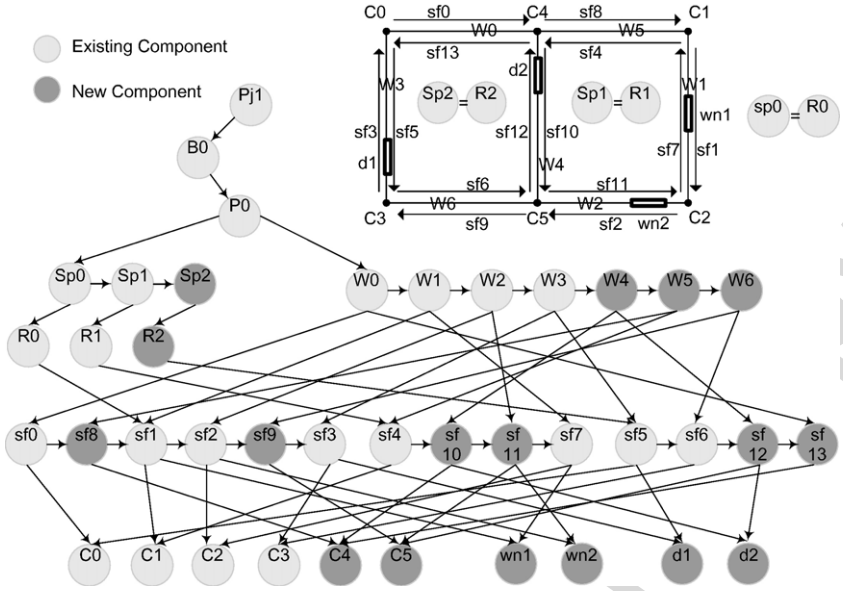


Fig. 5. Another simple example of the structured floor plan.

management algorithm for modifying such data in structured floor plan is explained in the next section.

5. The creation process and algorithm

5.1. Basic strategies in the process

It is important that the process should be done instantly and consistently. That is, the process should be activated at the very moment that the designer finishes a sequence of geometry inputs. An additional requirement is that it divides a set of input geometry given into several geometry units in order to simplify the input geometry. Considering the example shown in Fig. 6 where a simple existing room (ABCD) is going to be modified, the basic strategies are, therefore, as follows:

1) The input geometry can be composed of a set of several input lines that may or may not have intersections with the existing walls and even self-intersections with the input lines themselves. In Fig. 5 (upper part) 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 walls and self-intersections. As a result, each geometry unit, therefore, does not have any intersections. Fig. 5 (lower part) shows processes to divide the set of input geometry in to five geometry units.

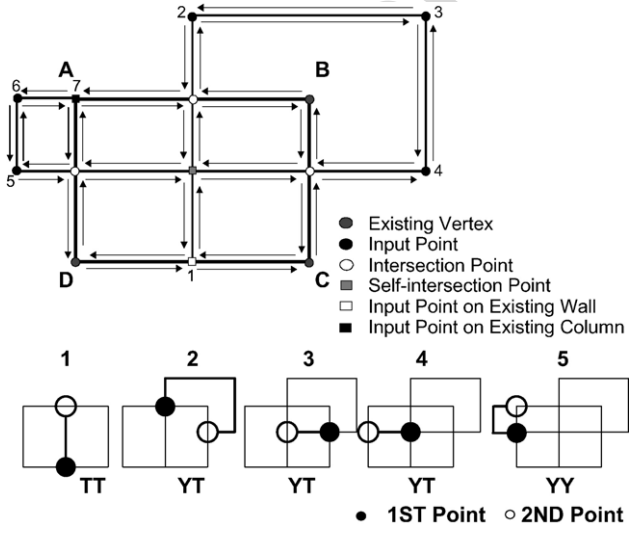


Fig. 6. The break-down process of the input geometry.

Table 1
Intersection types in a geometry unit

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

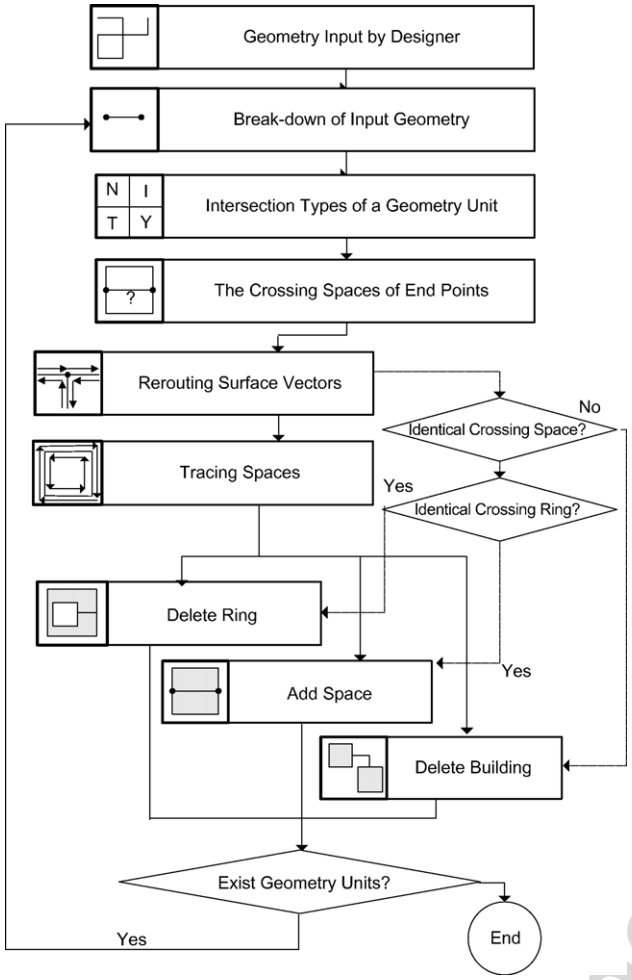


Fig. 7. The whole flow diagram of the creation algorithm for the structured floor plan.

4) There are four types of intersections with existing walls: T, Y, I and N. (see Table 1).
5) A special consideration is required in case that both ending points(a starting point and a target point) of a geometry unit are located on an identical existing wall.
6) A special consideration is also needed with inputs parallel and attached to existing walls.
7) Existing spaces (rings) in which input points are passing through should be recognized. A special concern is required when the existing spaces are different from each other.
8) Vectors of wall surfaces should be rerouted in a defined fashion based on the intersection type.
9) Entities for edited building components in the building data model should be corrected after tracing all existing spaces. Following is the description of the process in detail.

In summary, the overall spatial management algorithm for modifying such building data in structured floor plan is illustrated in Fig. 7. The details in each process are explained in the following section. The algorithm is evoked each time after the user has input and modified the architectural models. This creation algorithm assures a semantically rich and structurally correct floor plan at any point in the design process.

5.2. The break-down of the input geometry

Input geometry is a sequence of input lines 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 geometry units with no intersections at all.

The example in Fig. 6 presents complex input geometry on an existing structured floor plan. By tracing all intersection points, the break-down process divides the input geometry into five units. The following creation process is repeated five times to finish the whole processes.

5.3. Intersection types in a geometry unit

Each point in a geometry 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 is when the input point is located on an existing wall, while the input point of Type Y is on an existing column where two existing walls are crossing each other. In Type I the input point is placed on the end of an existing wall where two walls are not crossing. The input point on Type N is freely located on a space detached from any building components. All possible intersection types are displayed in Table 1.

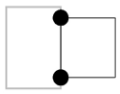
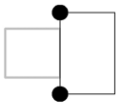
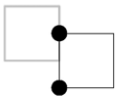
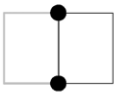
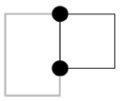
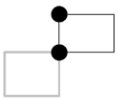
5.4. Input points on an identical wall

In order to perform the spatial modification correctly in the last step, special considerations are required when two end input points are placed on an identical existing wall. Nine input line types as shown in Table 2 can be found for three intersection types excluding N. Additionally, two more types for Type TT and Type YY should be considered by representing TT1, TT2, YY1 and YY2, respectively.

Table 2
Input line types on an identical wall

1ST \ 2ND	T	Y	I
T			
Y			
I			

Table 3
Input lines superimpose on an existing wall

In-In Type	Out-Out Type	In-Out Type
		
On-On Type	**On-In Type**	**On-Out Type**
		

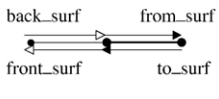
5.5. Input lines superimpose on an existing wall

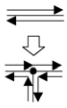
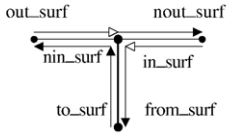
When an input line is superimposed on an existing wall, special care is needed. Table 3 presents six types of input lines on an existing wall in terms of an input point's location: whether it is on a column or wall or not. Input lines for type In–In, type On–On and type On–In can be simply ignored. In type Out–Out and type In–Out only input lines located at outside of the existing wall are considered as inputs. Particular care is not necessary for type On–Out.

5.6. Recognizing spaces crossed by two ending points

When a space is crossed by input geometry units, the space is needed to be divided. We call such space as a 'crossing space.' To find the space, it is necessary to find a wall surface faced to the input line in two wall surfaces of the existing selected wall. For Type T, calculating the triangle area is used to find the crossing space, whereas finding two wall surfaces closest to the input line guarantees to find the crossing space. Once the crossing space is recognized, it is traced to reroute wall surfaces and other entities properly.

Table 4
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;

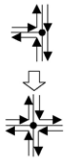
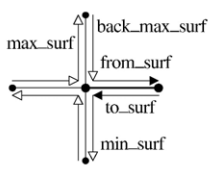
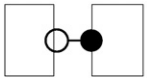
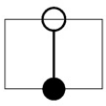
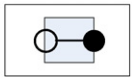
Type Y			to_surf->next = min_surf; back_max_surf->next = from_surf;

Table 5
Type of crossing space

Two spaces	One space	Two rings
		

5.7. Rerouting wall surface vectors

Once intersection types and the crossing spaces are recognized, vectors of wall surfaces connected to the input points are rerouted to maintain a correct structure. The rerouting process occurring in three intersection types excluding Type N can be found in Table 4. First, a relatively simple rerouting process occurs for Type I. When from_surf and to_surf are wall surfaces belonging to the input line, back_surf and to_surf are connected to from_surf and front_surf respectively. Next, the process for Type T creates one wall and two wall surfaces; nout_surf and nin_surf. Lastly, for Type Y it is important to find two vectors, one closest to the input line in clockwise direction and the other in counter-clockwise direction. To do this, it is necessary to calculate angles between each wall and the input line after collecting all surface vectors connected to the column.

5.8. Tracing spaces and re-structuring

As shown in Table 5, the type of crossing space is always one of three types: (1) Two different spaces crossed by two ending points; (2) A single space crossed by two ending points; (3) A single crossing space with different rings crossed by two ending points.

In the first case, input occurs between two different buildings, therefore two crossing spaces selected are outside spaces of two different buildings. After the process finishes, two buildings become one. The second is the most general case where one space is divided into two, creating a new space. The last is a special case

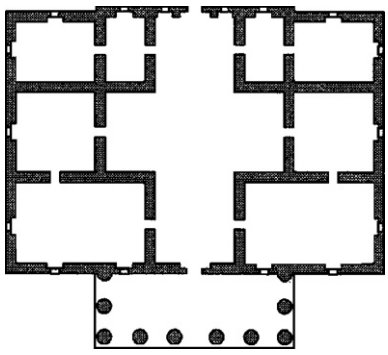


Fig. 8. Palladio's Villa Malcontenta.

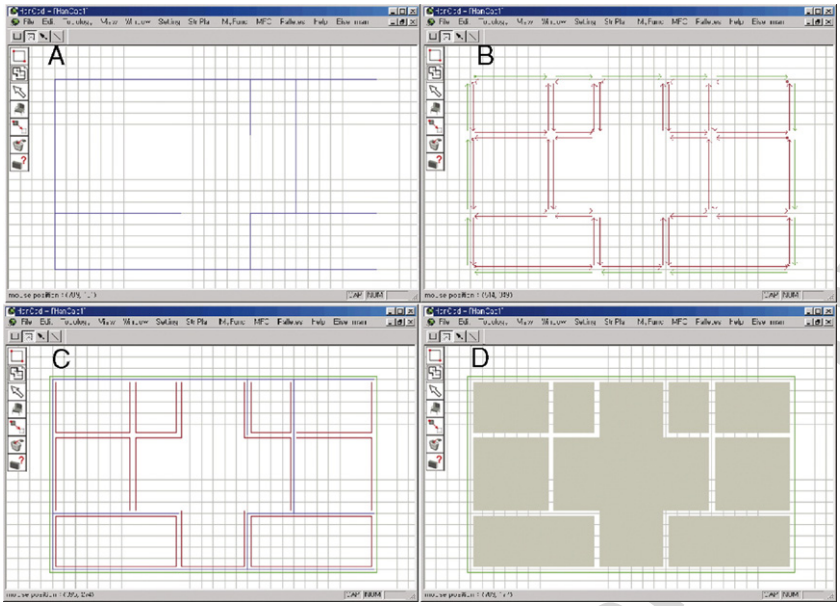


Fig. 9. Reconstructing Malcontenta on Str-PLAN and its multiple representations.

where a single space contains other spaces embedded inside. A space with a hole becomes a normal space without a hole. The system effectively supports this situation since it could have ring objects within a space object in the building data model of the system.

6. Case study

We developed a prototype CAD system, called Str-PLAN, using the creation algorithm described above. We used C/C++ for the programming language and Visual C++ compiler from MicroSoft Inc. for the compiling environment. It runs on IBM PC platforms with Windows 98/NT operating systems.

We selected Villa Malcontenta (Fig. 8) designed by Andrea Palladio, the great Renaissance architect to demonstrate and give an example. The building has been analyzed many times in various research in the architectural domain since this building, a well-known typical Palladian villa, is a clear structure of an appropriate size.

Fig. 9 shows the process of reconstructing the floor plan of Villa Malcontenta with Str-PLAN being developed. It not only 'draws' a floor plan as it looks, but also 'constructs' the building with design objects such as walls and columns. Spaces are automatically detected to represent a well-structured floor plan. With the floor plan, it is easy to build a 3D model since the floor plan also contains 3D building information. It presents the floor plan from various perspectives: A for the skeleton structure, B for the wall surfaces, C for the walls, and D for the spaces.

Fig. 9(B) visualizes wall surface vectors connected to each other in a specific fashion by constructing many rings that in turn represents rooms. As observed, all rooms consistently have surface vectors in the counter-clockwise direction, whereas an outside space keeps the clockwise direction. Two surface vectors exist for a single wall with opposite directions each other. Spatial relationships such as adjacency and window connectivity are automatically configured. This capability of managing spatial information could be used in further building simulations as such analysis modules are available. Str-PLAN can generate various drawings in different needs.

7. Application

The approach applied in this study, mainly based on a space-based floor plan representation of buildings, has a lot of

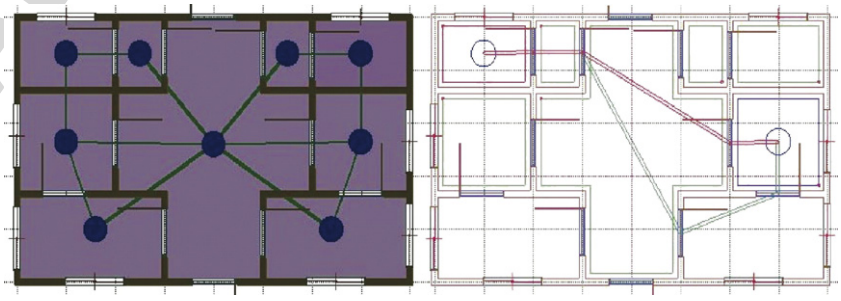


Fig. 10. (left) Adjacency diagram and (right) wayfinding in Str-PLAN.

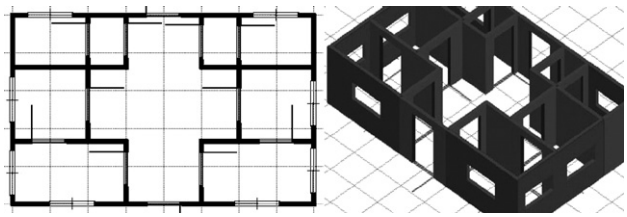


Fig. 11. Automated generation of 3D models.

potential application areas. Some of them are partly implemented and tested in the Str-PLAN environment, and others are under development.

7.1. Way-finding in the structured floor plan

O'Neill [9] attempted neural network simulation to predict way-finding performance in complex public facilities such as libraries, hospitals, and government buildings. The structured floor plan demonstrated an excellent capability to find paths based on the spatial networks automatically constructed during the design process as shown in Fig. 10. More detail exploration such as applying various way-finding criteria is needed.

7.2. Floor plan to 3D model

Since Str-PLAN is object-oriented and component-based, it constructs a floor plan along with 3D building information. Based on the information, the system can easily generate a 3D solid model as shown in Fig. 11. Beyond that, we are exploring a way of managing a floor plan and a 3D model simultaneously. Therefore, when a change is caused in a floor plan, another corresponding operation is transparently activated to adjust the 3D model.

7.3. Web-based approach

Current proliferation of the Internet provides an excellent infrastructure for collaborative design, an important aspect of architectural practices. In spite of the well-established infrastructure, software tools for design collaboration are rarely available and little research has been done to explore and develop such tools. As shown in Fig. 12, we are currently developing WebPlan, a web version of Str-PLAN by porting its C++ codes to Java language. We hope that the web-based CAD system can be used as an excellent collaborative architectural design environment upon its development.

7.4. Building compiler

The benefit of our systematical object-oriented building data model also applies to the concept of 'building compiler'. With the well-structured relationship among components and the real-time evaluation ability of structured floor plan, our novel system called 'Vitruvius Studio' is developed as a tool for evaluating physical requirements and spatial managements of apartment unit floor plans. [10] By appending building code in the system database to structured floor plan, the designer can 'compile' his/her intermediate design product to evaluate design errors during the design process. The compilation can be done immediately at any level or any time during the design process in a real-time manner. Fig. 13 presents the screenshots of Vitruvius Studio.

The evaluation system is composed of several modules such as a front-end component-based CAD engine, a knowledge base,

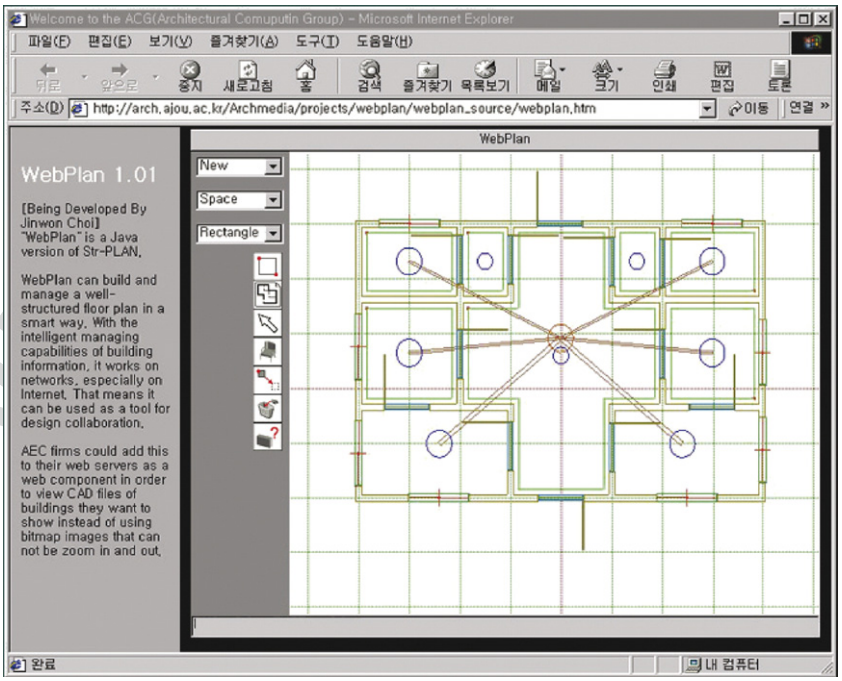


Fig. 12. WebPlan, a web version of Str-PLAN.

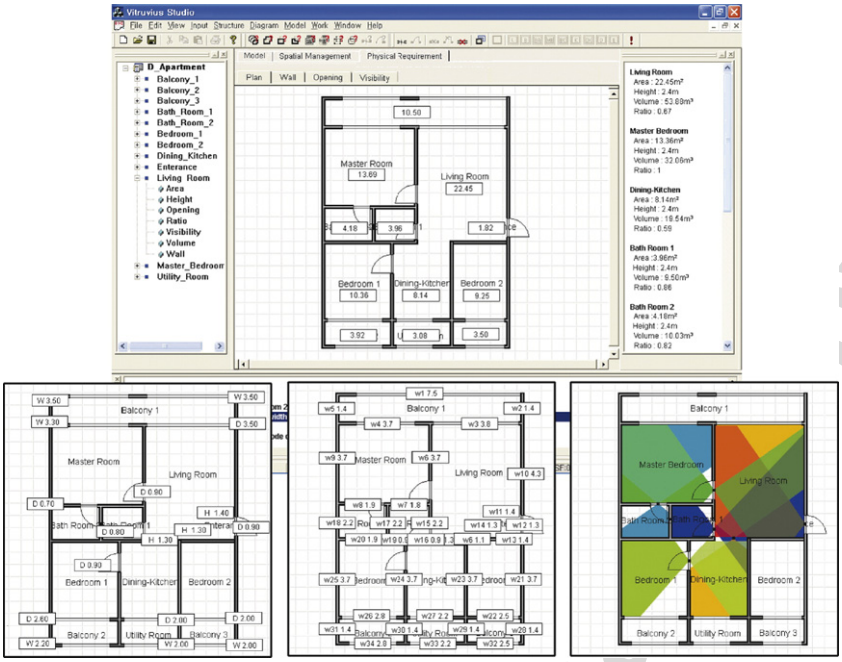


Fig. 13. 'Vitruvius Studio' based on Str-PLAN.

and a set of design agents. The notion of the design compiler is quite similar to a compiler for computer programming such as a C compiler.

7.5. Semantic location model

Another application of structured floor plan is the development of 'Semantic Location Model'. [5,6] The developed model embodies geometrical and topological information. The structured floor plan functions as the fundamental data structure suitable for applying the concept of context-aware system since all building components have been bound each other with their spatial relationships. Furthermore, our developed prototype called 'Vitruvius' capable of defining architectural spaces appropriately and manage them easily for constructing the ubiquitous computing environment. Given an inhabitant position, the system can trace and report the semantic location rendering the spatial relationships between the user and the space. For example, such information are the closest components, stared components and nearest exit as shown in Fig. 14. This context-aware system brings about the key component of the ubiquitous computing environment.

8. Conclusion

This study is the very first step for the development of a long-term large-scale intelligent CAD system. The building data model used in this project is based on a set of prior research on the building data model by the authors. As research progresses, the model will be further developed and sophisticated. The key issue of the study is to construct semantically-rich architectural

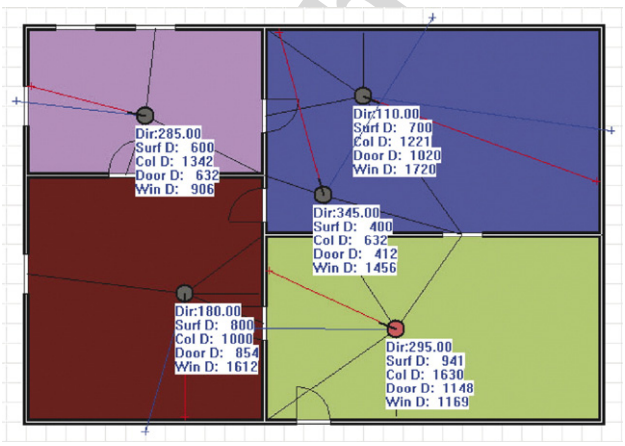


Fig. 14. 'Vitruvius' based on Semantic Location Model.

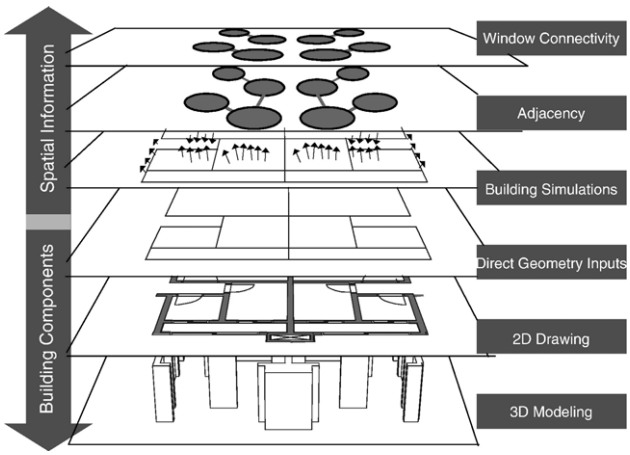


Fig. 15. Spatial and formal representations in Str-PLAN.

forms with minimum input from the designer in the interactive, instant, automatic manner. It is also an important issue to manage spatial design information as well as information about building components (Fig. 15). Managing both types of information, spatial and formal, is, we believe, very important for any intelligent architectural CAD systems. Other system requirements considered include automation and real-time construction/management in the process to construct a floor plan.

The future development of Str-PLAN will focus on tackling the following research issues: (1) powerful floor plan editing; (2) multi-story building representation by refining the building data model; (3) various analysis modules for analyzing and simulating building performances; (4) collaborative design environment that works on the Internet.

Acknowledgement

This research has been partially supported by the University IT Research Center Project in Korea.

References

[1] G. Carrara, Y. Kalay, Knowledge-based computational support for architectural design, Automation in Construction 3 (1994) 157–175.
[2] J.W. Choi, A development of an intelligent CAD engine to support architectural design collaboration, Korea CAD/CAM Journal (1997) 53–59.
[3] C.M. Eastman, Building Product Models: Computer Environments Supporting Design and Construction, CRC Press LLC, 1999.
[4] Y. Kalay, L. Khemlani, J.W. Choi, An integrated model to support distributed collaborative design of buildings, Automation in Construction 7 (1998) 177–188.
[5] Y. Lee, I.J. Lee, J.W. Choi, Location modeling for ubiquitous computing based on a spatial information management technology, Proceedings of CAADRIA 2004, Seoul, 2004, pp. 787–801.
[6] Y. Lee, J.W. Choi, J. Lertlakkhanakul, Developing a user location prediction model for ubiquitous computing, Proceedings of CAAD Futures 2005, Vienna, 2005, pp. 215–224.
[7] R. Liggett, A designer-automated algorithm partnership: an interactive graphic approach to facility layout, in: Y. Kalay (Ed.), Evaluating and Predicting Design Performance, John Wiley & Sons, 1992, pp. 101–124.
[8] A. Mahdavi, A comprehensive and computational environment for performance based reasoning in building design and evaluation, Automation in Construction 8 (1999) 427–435.
[9] M.J. O'Neill, Neural network simulation as a computer-aided design tool for predicting wayfinding performance, in: Y. Kalay (Ed.), Principles of Computer-Aided Design: evaluating and predicting design performance, John Wiley & Sons, Inc., 1992.
[10] J.W. Park, J.W. Choi, A computational approach to evaluate physical requirements and spatial managements of apartment unit floor plans, Journal of Asian Architecture and Building Engineering 2 (2003) b103–b109.
[11] C.I. Yessios, The computability of void architectural modeling, the computability of design, Proceedings of Symposium on Computer-Aided Design at Suny Buffalo, New York, 1986.
[12] C.I. Yessios, Architectural modeling, Architecture 844 lecture notes I, Graduate Program in Computer-Aided Architectural Design, Department of Architecture, The Ohio State University, 1986.