ARCHITECTURAL RESEARCH, Vol. 1, No. 1 (1999), pp. 23-30

A House Design Automation System Based on the "Design-by-Novice" Paradigm

Uk Kim
Hongik University, Seoul, Korea

Jinwon Choi
Ajou University, Kyunggido, Korea

SungAh Kim
MyongJi University, Kyunggido, Korea

Abstract

This research investigates a system for house design automation. The system is based on an object-oriented building data model, aiming to support the house design process conducted by non-expert users. Its object model, with simple yet powerful user interfaces, enables a CAD system to handle a complicated building system with much ease. Hence, the model dramatically simplifies the design process beyond just the automatic document generation. In this paper, we discuss the aspects of the building data model, introduce critical concepts such as grid objects and structured floor plan, and present a prototype system called GPLAN. The system is implemented in the framework of our building data model, and it provides a host of intelligent features that have been proved useful for house design automation.

Keywords : Design Automation, Object-Oriented Building Data Model, Grid Designer, Structured Floor Plan

1. INTRODUCTION

The dissemination of quality housing design in the form of electronic design documents has been one of the most important services performed by the Ministry of Agriculture in Korea. The goal is to raise the living standard of the farming population by providing a standardized and improved housing design service at nominal costs. During the last five years or so, several prototype models have been proposed so that the service providers could conduct with more efficiency. The documents such as construction drawings and specifications have been generated using CAD software in advance, and then provided to the user as a ready-made set. Thus the users, i.e. farmers, could choose the best-suited design with help from government agents. While the prototypes proved to be successful in terms of construction efficiency, both the service providers and end users have often requested better functionality from the current CAD software.

The first problem is that both the service providers and the served usually do not have enough architectural literacy or design expertise. Hence, it is not easy at all for the users to modify the drawings or documents when they want to change the given design. In fact, it is rare that the prototype design is used as provided. The user needs to operate the CAD software even for minor changes. The existing CAD software is too complex to learn in depth, and the maintenance cost is relatively high. On the other hand, even a minor modification requires a thorough change over all of the related documents. CAD-generated documents are not far better than the traditional ones when it comes to the automatic propagation of design changes.

Even after the change, modifications rarely enter the computer, thus the next similar service cannot take advantage of the precedent projects. This caused the disintegration of the project data, leading to very inefficient design service. Such problems call for a new design documentation using a simple yet flexible design software. The following are main features to be considered in the new system:

- Be able to quickly generate any type of housing plan with ease even by novices.
- Be able to edit prototype plans so that individual design needs can be best fulfilled.
- Any modification and changes in the prototype models should be propagated in an automatic fashion to other drawings and other design documents, and vice versa.
- Be able to generate necessary drawings and documents including automatic speci-

* This material is based upon work supported by the Agricultural R&D Promotion Center under the Ministry of Agriculture and Forestry

fication.
- Any generated design documents should be accessible through the Internet service which should be managed in the form of design project-database in the long run.

2. ISSUES TO CONSIDER

We have focused on the following issues to be considered in our prototype:

Tentative Nature of the Design Process - The existing CAD software, most developed to represent what is precisely known and to analyze what is precisely defined, is insufficient to support the tentative nature of the design process. Complex geometries seem to take too long to input to the computer, and they are too cumbersome to modify quickly. Creation of design objects should be flexible and easy enough so that the user can generate and evaluate various design alternatives as quickly as possible. The tentative nature can be considered less critical at this moment as the system is mainly used for creating designs by modifying existing models. In fact, the initial requirements of the conventional design system mainly focus on the automation of document generation. However, we address this issue considering the potential of the system to be an extensive design tool in the long run.

Simplicity - Designers should not be overwhelmed by a host of low-level object construction tools. The design strategy of the GPLAN system should reflect this strategy so that a design can proceed only with high-level commands if possible. Hence, the system provides only a few set of commands, mainly for sketching walls, inserting components, and manipulating views. They are well integrated so that a designer is provided with the flexible representation of evolving design while he uses the tools with the aid of intelligent object tools.

Reusability - The reuse of well-tested and optimized design projects allows reducing design times, improving design quality and the predictability of the designs. The reuse ranges from selecting building components from a library to adapting an already designed object to the specific requirements. Many CAD tools claim that they provide reusable design components, but most of them understand the reusability in terms of reproducibility. The object-oriented building model of our prototype system increases the reusability of the precedent design by allowing for flexible modification while maintaining the consistency of the database.

Partial Modification - A specific component of a design object can be easily modified. When a partial modification takes place, the effect of the change should be propagated to other objects when needed. Any modification leads to the change in the design database as well. The versioning process should take place transparently so that the designers need not concern themselves with version control.

Levels of Detail - When using computer-aided design tools, a designer is incessantly distracted by a variety of aspects such as design alternatives, excessive details and organizational procedures. A proper level of detail is provided so that the designer is not overwhelmed by the details of the represented object. Ideally, the CAD system should be able to support different views where the abstraction in effect increases or decreases the level of detail.

Multiple Representations - The system must handle multiple object views in a consistent and convenient fashion. This requirement is akin to the requirement for support for level-of-detail and abstraction hierarchies. A designer should be able to manipulate an identical design object while he/she 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 interests. Different types of representation can be mixed up 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 information delivery.

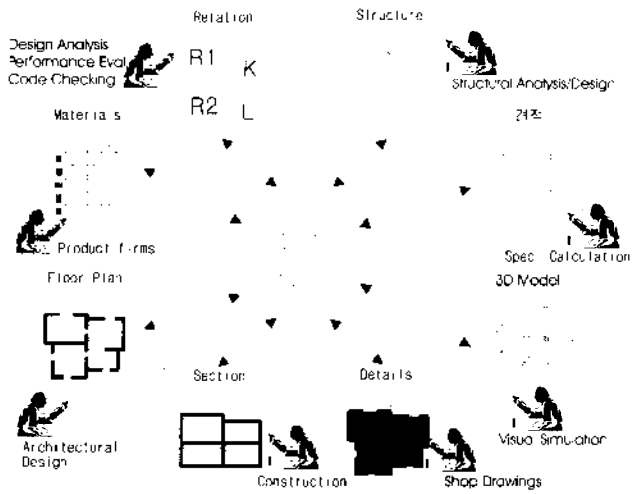

Figure 1. Multiple representations and design collaborations based on a single building date model

3. BUILDING DATA MODEL

In order to meet the required features, a building data model of house prototypes was defined and implemented using the object-oriented design method (Figure 2). Developing the building data model has been an especially rigorous part of this research because of the inherent complexity of the building system. Each component of a house is in fact an intertwining of structural elements and spatial relations.

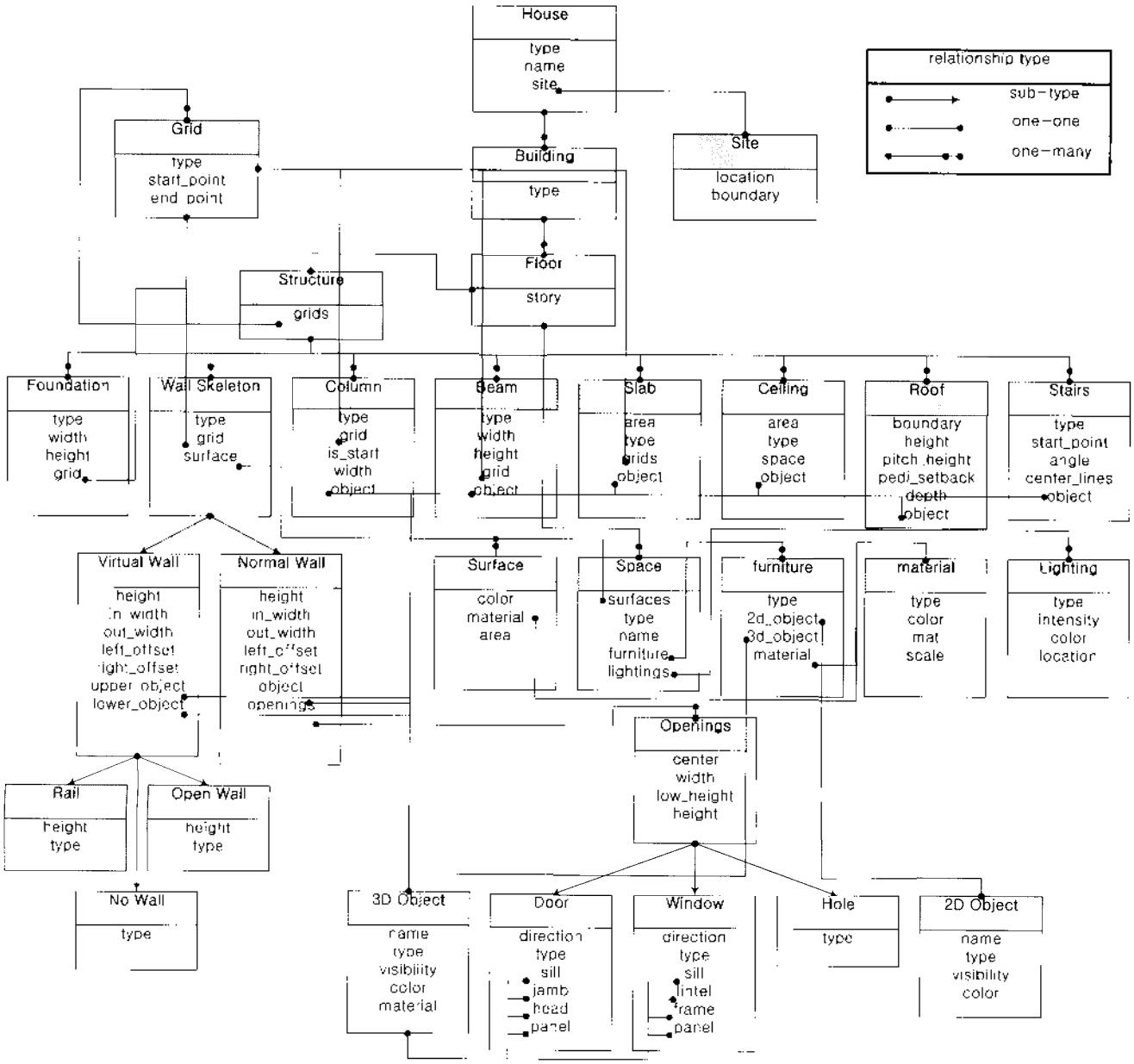

Figure 2. The GPLAN building data model

While we cannot accommodate every future application's needs within a single integrated model, we succeed in establishing a common conceptual foundation through which a variety of data models can eventually communicate. Consequently, the building model allows the system to command a variety of advanced object-oriented features such as multiple representations, levels of detail and flexible editing while maintaining the consistency of the design database on its fundamental level.

The development of a building data model requires a division of the design process into higher level design operations and lower level database integrity/maintenance operations. The reason for this division is not so much to simplify or clarify the complexities of the design process, but simply to isolate and extract those aspects that can be incorporated into an integral component of a design database. The distinction between design operations and database maintenance operations is by no means an obvious one, since the propagation of changes initiated by design operations may lead to conflicts that can only be resolved by other design operations. This distinction, however, forms a basis for both defining and developing a building data model.

Currently, our building data model lacks compatibility with other mainstream data models. The next phase of the research will focus on this aspect more in depth.

4. CONSTRUCTING WELL-STRUCTURED FLOOR PLANS

UNIFORM – A Structured Floor Plan Editor

The concept of structured floor plan is introduced along the grid object. There has been some research on developing CAD systems that can build structured floor plans (Yessios, 1986a & 1986b; Kalay et al, 1995; Choi, 1997). In such systems, a floor plan is structurally well defined by having a hierarchical structure of components. Since space and form are two main aspects in describing a building, they should be represented together in a system. To be more effective, this can take an object-oriented approach, where each building component is an object from the view of the object-oriented paradigm. The object has its own data and methods of how to behave in certain situations.

Figure 3 shows a typical user interface of UNIFORM, a structured plan editor implemented on the Macintosh platform. Having capabilities to create and manage only a well-structured floor plan is not enough to be a design tool. How to create such a floor plan is also an important issue to be addressed. That is, the way to create a floor plan should be as natural and flexible as possible. The user can start from a simple rectangular shape and develop a very complicated floor plan by inserting and deleting lines or polygons in many different manners. Among several building components including walls, columns, doors, windows, slabs, roofs and stairs, the primary components, essential to defining a basic building geometry, are the walls and columns. Other components are considered secondary.

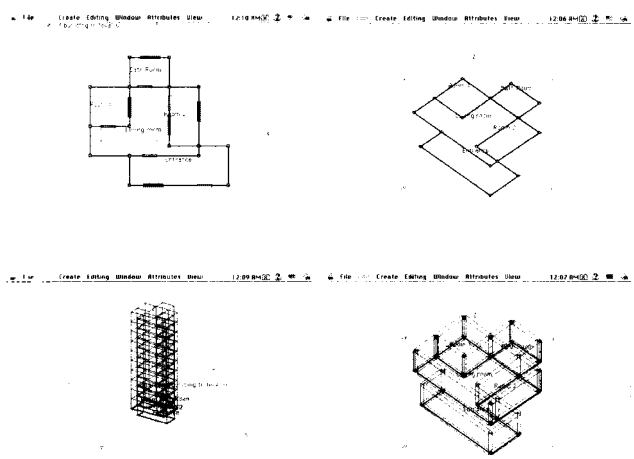

Figure 3. The original Macintosh version of UNIFORM – Structured Floor Plan Editor

GRID Object as a Schematic Plan Designer

The Grid system as a control mechanism of the housing plan has been introduced to the system. Thus, the designer is able to generate or modify basic housing plans with ease by defining simple grids. The defined grids then are used for detecting spaces and creating structural components of the house, all of which are managed by the system's object manager. The grid operation works on a tool called "Structured Plan Editor" which secures the integrity of the space-structure relations in a design database. More features have been integrated making use of the object-oriented design technology, e.g. multiple-representation, level of detail, and so on.

A space is, for example, confined by the surrounding walls. We can think of this kind of space as a room. The room can be assigned a variety of functions, such as a living room, dining room, and so on. Designers usually initiate the spatial configuration by generating a set of lines. These lines can be actual walls or conceptual perimeters bounding a functional space. It is difficult for a designer to determine the character of each line from the beginning. When the CAD tool is used in this process, the design constraints become more apparent. Current object-based CAD tools usually force users to input the actual building objects from the beginning, which is not the case in most actual design activities.

Thus, we introduced the system of grid. The designer generates various design schematic plans using grid objects.

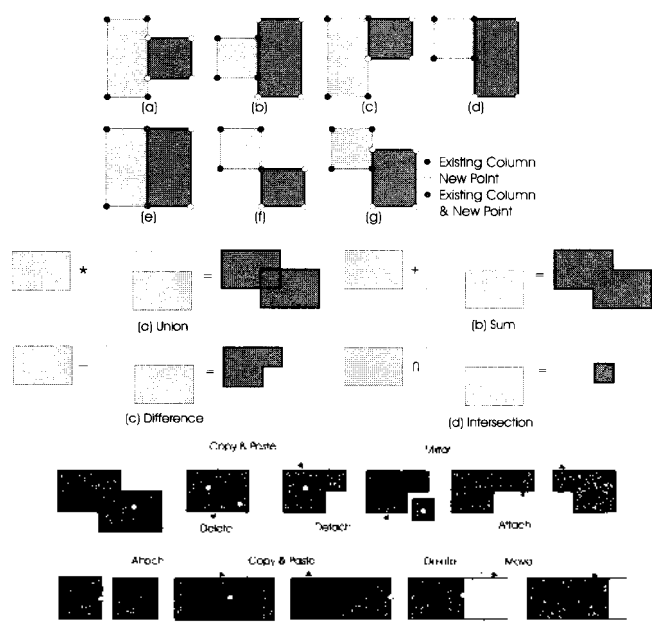

Figure 4. Schematic design operations using GRID objects

5. FEATURES OF THE GPLAN SYSTEM

System Overview

We have tested our model using our prototype implementation GPLAN. GPLAN allows the user to quickly generate schematic floor plans using the grid tool. Then, it automatically detects the spatial configuration and defines the structural relations based on a network of design objects. In addition, it provides a set of intelligent design tools for refining the design. This implies that the user is able to create as many alternative designs as possible in a limited amount of time. Its intelligent multiple representation mechanism also allows the user to control the suitable level of detail with ease.

Implementation

The current version of the GPLAN system is implemented as a house design tool. Next releases will incorporate more classes of building design models such as a multi-story apartment complex. The system runs on Windows machines. It is implemented on dual platforms, an MFC-based stand-alone application built with its own 2D/3D CAD engine, called HANMOD, and an AutoCAD-based application. In the latter case, AutoCAD™ software is used as the graphic front-end in which generating objects and high-level design operations are conducted. C++ along with ObjectARX™, an object-oriented application programming interface of the AutoCAD software, is the primary implementation language.

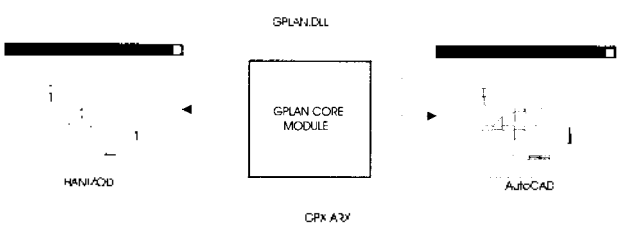

Figure 5. Dual platform applications of GPLAN

Main Features

Rapid schematization with the grid tool - The system provides a set of grid tools in terms of geometric characters. For example, the user is able to draw the grid lines by invoking a single grid line, polygons, rectangles, or arcs. It looks more like drawing the center lines of walls. As the grid objects are drawn, the system automatically detects the intersections, dividing the grid lines into pieces. After the necessary grid lines are generated, wall skeletons are placed to draw walls. Intersections among skeletons are properly considered to resolve wall joints (Figure 7). 3D walls are automatically generated at the same time. Openings are also placed on walls. They can be moved within a wall or to another wall. Since a wall component is linked to other adjacent walls, the user can simply pick and move a wall without worrying about wall connections, openings within it, or even 3D objects. The system takes care of the cumbersome jobs automatically.

Levels of abstraction - The user has mainly three modes of representation. The Skeleton mode displays only the skeletal images of the wall. This is the most abstract representation of the model. Design operations on specific walls are much easier as the user is free from unnecessary details of the drawing. In this mode, most of components, e.g. the windows and doors, are represented as boundary boxes, while the wall shows a set of outlines. The 2D-Draw mode displays the two dimensional symbolic representation of the drawing. The level of detail can be controlled more in detail even in this mode. Lastly, the 3D-Model mode shows the three dimensional solid model of the house, with which photo-realistic visualization can be conducted.

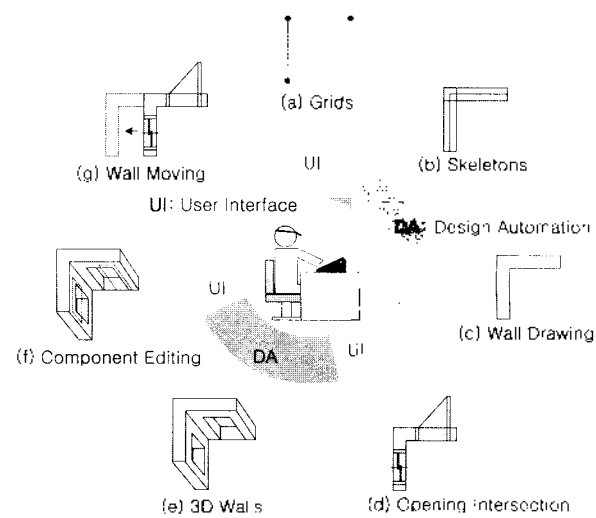

Figure 6. User-interaction and design automation in the design process

Flexible editing – The generated grids are converted into walls by the 2D-Draw command. Each wall is assigned default properties: solid brick wall with predefined physical dimensions. A set of interactive editing tools are provided so that the user can pick any wall and edit its properties. The system keeps maintaining the consistency of the floor plan structure. The modifications of the properties are reflected instantly on the 2D-Draw and 3D-Model representation modes. Modifications of materials, dimension, and insulation types are currently supported. One interesting wall type is the "virtual wall" which does not actually exist. Such a wall type is needed to define a boundary between two distinct functional spaces. Designers often carelessly draw lines at the schematic design phase, but the system needs to differentiate the

spatial functions explicitly. The virtual wall is useful in these cases.

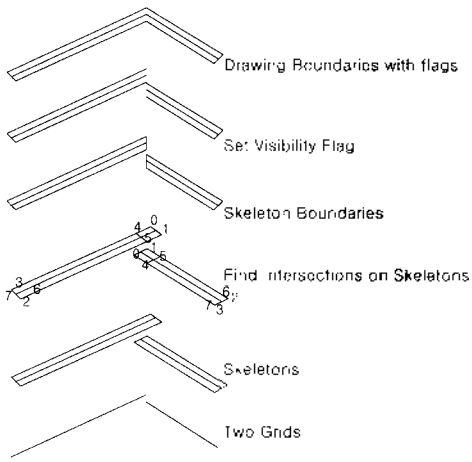

Figure 7. Resolving wall joints using grids and skeletons

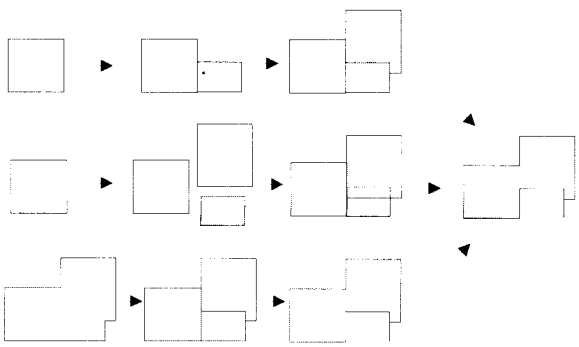

Figure 8. The flexible editing process with the GRID tool

Automatic space detection - Recognizing spaces is an important capability of the system. Spatial information is used for constructing slabs, ceilings and roofs. The system can generate spatial adjacency matrices, and the matrices are, in turn, used for checking spatial relationships at the very beginning of the design process. Figure 9 shows how to detect spaces. Since skeletons placed on the top of grids have points in the clockwise order, tracing the skeleton points allows the system to detect spaces. The example in Fig 9 shows three spaces detected. A space is represented by the list of wall numbers and wall surface numbers. The outside space following outside walls is numbered clockwise while the other inner spaces counterclockwise.

Document automation - Required drawings are automatically generated. After the user builds the schematic floor plans, the system can generate drawings: floor plans, elevations, sections, and 3D model views. The current system still lacks enough detail generation functions. The next release will be able to generate other technical documents such as construction specifications and mechanical drawings. Another feature for the document automation is a roof design method. Both laymen and designers easily place a pitched roof with parameterized pitched lines and perimeters, by selecting rooms on which the roof is covered.

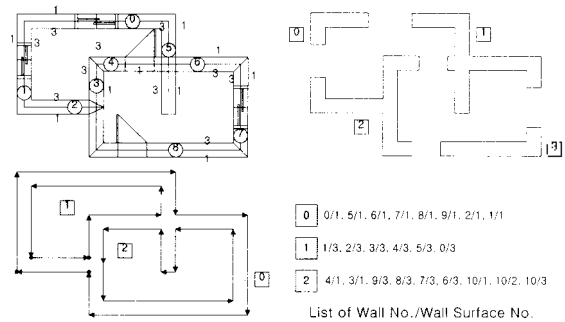

Figure 9. Detecting spaces and wall drawing

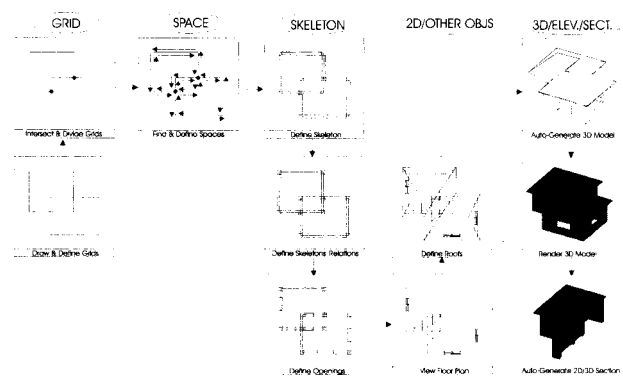

Figure 10. The house design process using the GPLAN system

6. CONCLUSIONS

The house prototype design system implemented for this research is a modeling tool, with the capability of integrating two and three dimensional information of farming house building components. The model consists of such components as spatial elements, structural elements and building envelopes. The objectives of this system are twofold; to help farmers choose suitable building types conveniently while allowing design changes in real time with automatic documentation, and to provide designers with an enhanced design tool. This research is a three-year project, and has just finished the first year work. So far, all the necessary objects are defined to shape a house prototype. Most objects are generated through a graphical

interface using a commercial CAD engine with object managing extensions, e.g. AutoCAD R14.

One intelligent feature of the system is that spaces are automatically detected by boundary walls. To be consistent with the space property of the boundary condition, virtual or imaginary walls are introduced when there exist spaces separated without physical partitions. Also, to adapt to the tentative nature of the early design stage, grid objects are manipulated as the primary spatial design element for space layout. Walls cling to grid objects, and thus when space shapes are changed, users just edit the grid objects with wall properties.

In order to complete the prototype house design system, the following aspects, other than the completion of automatic documentation, are being developed and implemented:

1. Which design components can be modifiable at which stage to which degree of freedom. Designers can possibly weigh the properties of the specification.
2. How can the system retrieve the most suitable plan type the user can start off with? The system must search project specifications for the most suitable module in a design database for given requirements.
3. After the modeling and documentation functions are completed, how can data exchange or sharing with other domains be accomplished for design analysis and evaluation? Compatibility with STEP or IFC data models may be an answer.

By solving the above research issues and experiencing implementation technology and the development skills, the integrated building modeling and design system can be apprehended for practice in design.

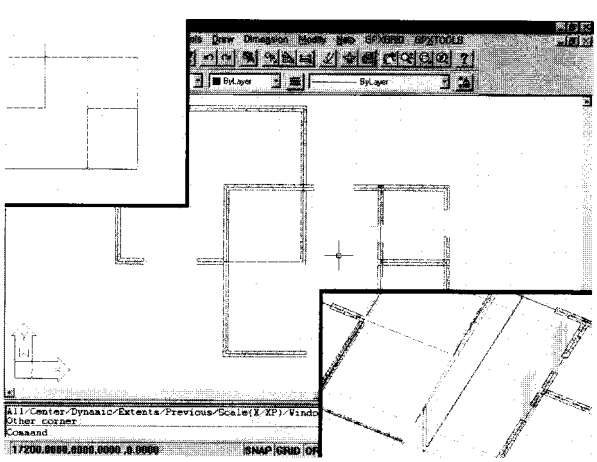

Figure 11. A collage image showing the design process using the GPLAN-AutoCAD version

REFERENCES

Choi Jinwon (1997) A Development of an Intelligent CAD Engine to Support Architectural Design Collaboration, Korea CAD/CAM Journal, pp. 53-59,

Cornick, T. (1996) *Computer-Integrated Building Design*, E & FN Spon, London, UK

Coyne, Roseman, Radford, Balachandran, Gero, (1990) *Knowledge-Based Design Systems*, Addision Wesley, NY

Cross, N. (1977) *The Automated Architect*, Pion Press, New York

Eastman, C.M. (1988) Automatic Composition in Design, *Design Theory and Methodology*, Newsome, S. *et al.* (eds), Springer, New York,

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

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

Eastman, C.M. *et. al.* (1993) System Architecture for Computer Integration of Design and Construction Knowledge, *Automation in Construction*, 2, pp.95-107

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

Harfmann, A. Chen, (1990) Building Representation within a Component Based Paradigm, *ACADIA Proceedings '90*, pp.117-127

IAI, International Alliance for Interoperability (URL=http://iaiweb.lbl.gov).

Kalay, Y. E., Khemlani, L. & Choi, J. (1998) An integrated model to support distributed collaborative design of buildings. *Automation in Construction*, Elsevier, 7, pp.177-188, 1998.03

Kim Uk, (1992) An Object-based Modeling System Coupled with Design Information System, *6th International Conference on Systems Research*, Baden Baden, Germany

Kim, Uk, (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.,

Koutamanis, A. (1993) "The Future of Visual Design Representations in Architecture", in: Automation in Construction, 2 p.p.47-56

Mitchell, W. J. (1977) *Computer-Aided Architectural Design*, Van Nostrand Reinhold, NY

Pohl, J., Myers, L.A. (1994) Distributed Cooperative Model for Architectural Design, *Knowledge-Based Computer-Aided Architectural Design* (G. Carrara & Y.E. Kalay, eds.), Elsevier, Amsterdam

Pollalis, S.N. (1994) Towards the Automation of Conceptual Design, Tzonis, A., Pollalis, S.N.,

White, I. (eds.), *Automation Based Creative Design: Current Issues in Computers & Architecture*, Elsevier, Amsterdam

Tello, E.R. (1989) *Object-Oriented Programming for Architectural Intelligence*, Addison-Wesley, NY

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