Design Evaluator:

Critiquing Freehand Sketches

Yeonjoo Oh

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Architecture

University of Washington

2004

Program Authorized to Offer Degree: Architecture

University of Washington
Graduate School

This is to certify that I have examined this copy of a master's thesis by

Yeonjoo Oh

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Mark D. Gross

_____

Ellen Yi-Luen Do

Date: _____

Signature_____

Date _____

University of Washington

Abstract

Design Evaluator:
Critiquing Freehand Sketches

Yeonjoo Oh

Chairs of the Supervisory Committee
Professor Mark D. Gross/ Ellen Yi-Luen Do
Department of Architecture

This thesis concerns about how feedback integrates into a sketch design system. Design Feedback as critiques can help the designer better envisage what a design will be like in advance, allowing the designer to avoid certain kinds of errors. This thesis discusses how a freehand sketch critiquing system can be developed and how this tool can support in the architectural design process as design tool.

The thesis proposes a working prototype, Design Evaluator to demonstrate the potential of this knowledge-based design system. The Design Evaluator system has the design knowledge translated into rules related to four architectural spatial issues: proper room sequence, adjacency, room placement, and minimum area.

The Design Evaluator interprets the designer's architectural diagram and recognizes the spatial relations such as circulation paths and room placements. It checks the architectural diagram with the built-in rules. When a rule violation occurs, the system displays the design critiques in three ways: text messages, annotated drawings, and texture-mapped 3D visualization. These critiques stimulate the designer's 'reflection-in-action' cycle during the sketching of her/his design ideas. Moreover, they help the designer to detect errors in the early design stage.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acknowledgements

I am grateful to the following people: my thesis committee, colleagues, and family.

Especially, I want to express my gratitude to my thesis committee, Professors Ellen Yi-Luen Do and Mark D. Gross. They were always there for me and gave me many extremely helpful comments. I've learned many things from them: from the thinking process to the knowledge of the design computing field. I will never forget my thesis presentation week. Everyday they read my presentation material, heard my presentation and gave me great comments. They have been a constant encouragement to me throughout the whole process.

I have been a participant in a terrific working community, the Design Machine Group. I thank everyone in the DMG, especially Chen Je Huang, Doo Young Kwon, Eunsoo Lee, Karen Hanson, Markus Eng Ken Camarata, and Mike Weller. Karen, especially, helped me with my writing. While preparing for my thesis presentation, Ken, Mike, and Markus heard my presentation several times and offered suggestions to improve my presentation.

I am so grateful to my family: mother Soyoung Lee Oh, father Yongmu Oh, and my brother Youkeun Oh. I thank them for encouraging me to follow my interests and giving me unconditional support. I especially want to express my gratitude to my mother, who she called me everyday to listen about my everyday life. I also want to thank my best friend, Minjin for listening to my complaints and frustrations.

This research was supported in part by the National Science Foundation under Grant CCLI-0127579. The views and findings contained in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Chapter 1

Why Design Evaluator

1.1 MOTIVATION – AVOIDING DESIGN ERRORS

Designers, even expert designers make errors. These errors though they may be small affect the whole design process and the end-product. If these errors are not caught in the early design stages, it is time-consuming and costly to revise and correct them.

Design errors often happen as the result of cognitive limitations or the overlook of design knowledge. During the design process, designers have the cognitive load of processing and managing massive amount of information such as functional constraints and specification requirements (Tversky, 2002; Lee, Eastman, and Zimring, 2003). In addition, occasionally it is possible that the designer actually may overlook some design knowledge. That is critical for some task. The design knowledge can help the designer better predict or envisage what a design will be like in advance, allowing him or her to avoid certain kinds of errors.

Design is an exploration of the constraints and alternatives (Gross, 1986). Constraints include design rules, relations, rules of thumb and conventions to be maintained in design. The designers work within the constraints that they have. For example, the hospital design constraints include "The emergency room should be in the clinical area" or "The ward should be 40,000 square feet, considering the planned capacity for patients and the sizes of several necessary pieces of furniture." However, the designer very often violates these constraints. For example, if the designer designs a smaller ward than the constraints would allow (40,000 square feet), this small error affects the whole design, and may require rearranging or resizing the rooms. It is because

design decisions are dependent upon each other.

We have to understand that the constraints often relate to each other like the patient capacity and the size of ward. Therefore, researchers often address these constraints by analyzing the design in multiple ways. Sometimes these are too complex to be described by the rules (Ishizaki, 2002). Nevertheless, some designers are able to reduce errors by applying heuristic methods such as the rules to their designs (Lee, Eastman, and Zimring, 2003). Rule-based description can be more accessible to use and much easier to write than other description methods like a case-based system. For implementing an intelligent design system, the system should have the knowledge related to the architectural design. Providing the designer with the related design knowledge through the true-false test is much easier than finding and giving the previous cases. It is due to the character of architectural design. Architecture design handles the space, so it has really dynamic character. Therefore, describing the cases of architectural design and finding the proper cases are much more difficult.

These rules, when presented to designers in the form of knowledge advice, can lighten the cognitive load imposed by the design problems. Rule-based computational design systems offer the opportunity to help the designer avoid errors. These rules can be provided in the form of feedback or critiquing. This type of computational design system can check the proposed design and provide the designer with the feedback of rules that are violated in the design. With this feedback the designers can avoid design errors and take advantage of unseen opportunities.

1.2 ARCHITECTURAL REASONING ILLUSTRATED BY STEVEN HOLL'S SKETCHES

Architectural drawings are symbolic representations. They reduce the cognitive load of keeping in mind the spatial arrangement of a building and what activity is assigned to each space.

Drawings are a record of the designers' reasoning. Designers record their ideas and concerns on their drawings. Upon careful examination of a design drawing, one can identify the designer's reasoning about issues. For example, architect Michael Graves describes that he sketches to record his observations and discoveries. His shorthand notes and sketches are kept, and often changed or combined with other versions of sketches. He also states that the symbols represented in his drawings are a kind of language to communicate with himself or others (Graves, 1977). Therefore, one can get insights into architects' ideas and reasoning process by carefully examining their sketches. For example, below is our attempt to decipher the reasoning process in a few of architect Steven Holl's design drawings.

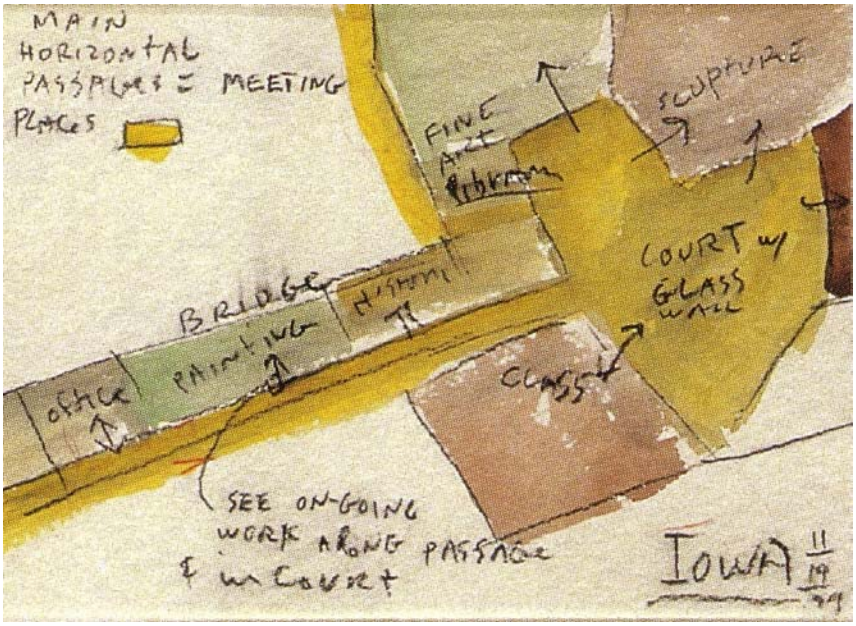


Figure 1.1 Visual symbols in Steven Holl's design drawings for the University of Iowa's Art and Art History Building include wall lines and text labeled spaces: The circulation path (passage way) is highlighted in yellow. Double-headed arrows indicate visual access. (Source: El Croquis, Holl, Steven, 2002)

Figure 1.1 is an early design drawing by architect Steven Holl for the University of Iowa's Art

and Art History Building (Holl, 1999). In this drawing he used lines and arrows to represent walls and visual access. He also wrote “office”, “painting”, “history”, “class”, “court”, and “sculpture”, etc. to label these functional spaces. He wrote "main horizontal passages = meeting places" with a yellow box as a legend, and drew the pedestrian circulation passage in yellow. Several double-headed arrows indicate visual access between the passage and the classrooms. A call-out arrow from the path is linked to the text of "see ongoing work along passage in court". These graphic symbols and text annotations indicate that the designer is concerned about the passageway between the court and the other classrooms (Figure 1.1). He considers visual access, material, and functional arrangement in the simple design drawing. He used graphical elements (such as lines, arrows and colors) and shorthand notes for recording his ideas.

1.2.1 Spatial Concerns

Architects see spatial relations such as connection and adjacency among the spaces in their drawings. In the example (Figure 1.1), a “court” (polygon space on the right) is connected with a sculpture room (top right) and a classroom (lower left). These spaces are clearly labeled “sculpture” and “class.” The architect has written, “w/ glass wall” below the functional label of “court” to note a material choice. Arrows from the court to sculpture room indicate a concern with visual concern (i.e. where people can see the sculptures through the glass walls).

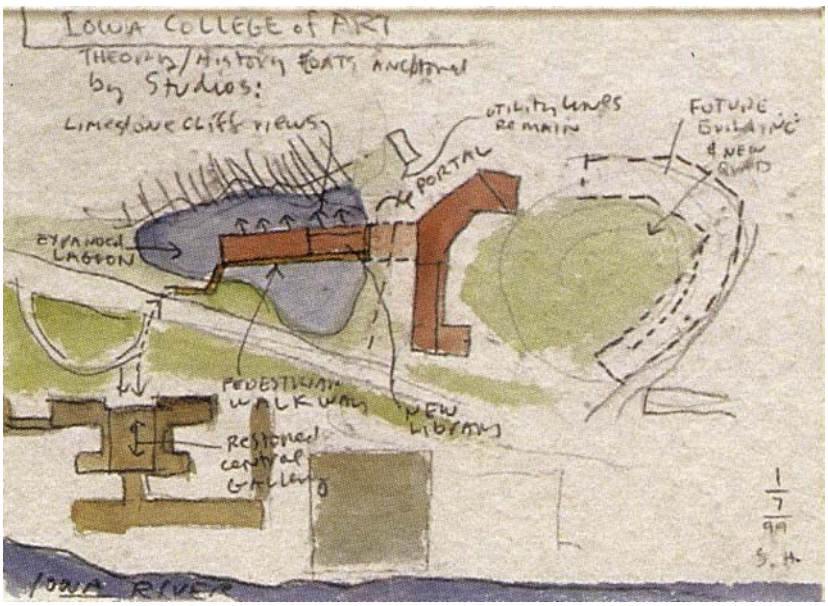


Figure1.2 The different functional spaces are drawn in different colors in the concept sketch for the University of Iowa's Art and Art History Building (Source: El Croquis, Holl, 2002)

Architects also use the drawing as a medium to contemplate spatial arrangements. For example, Figure 1.2 shows a concept sketch in which the different colored shapes represent different functional spaces. The main school building with classrooms and a library is colored in dark red (top). The brown connecting rectangles represent a gallery building. The coloring of the spaces makes the focus and concerns more visible on the paper and perhaps helps the designer to remember the idea or to communicate it with others. Arrows from the main school building and notes "limestone cliff views" represent his concerns about visual access and views issues.

1.2.2 Functional Concerns

One can also identify architects' concerns and decisions about functional arrangement of spaces and circulation from their design drawings. For example, in the plan for a small residence, Y House (Holl, 1999), Figure 1.3, Holl wrote "MBR", "BR", "DR/K" and "LR" as functional labels. The connecting linear shapes in yellow (center of the drawing) represent a continuous ramp. We

can see that the designer drew a call-out line to label this as a “Y” ramp. The rectangle symbol next to the ramp represents a staircase. It appeared that this is a design for a two story house, judging from symbols (stair and ramp) and text (“upper level” and “below”). In this drawing, the designer is concerned among the functional arrangements about the different floors. For example, on the top right, the architect wrote “BR below LR”, to indicate the placement of a bedroom below the living room (at this level). On the lower part of the building, similar markings of “MBR (master bedroom)” and “BR (bedroom)” also appear on the drawing. Adjacent to the rooms there is an arrow with the text “DR/K” (Dining room /Kitchen). Holl also circled his annotation of “2BR upper level” (lower left). This drawing shows that the designer was concerned with functional space arrangements and spatial relationships such as horizontal or vertical adjacency between the rooms. Each functional concern was carefully annotated with the label of the space and some also with notes about the spatial arrangement.

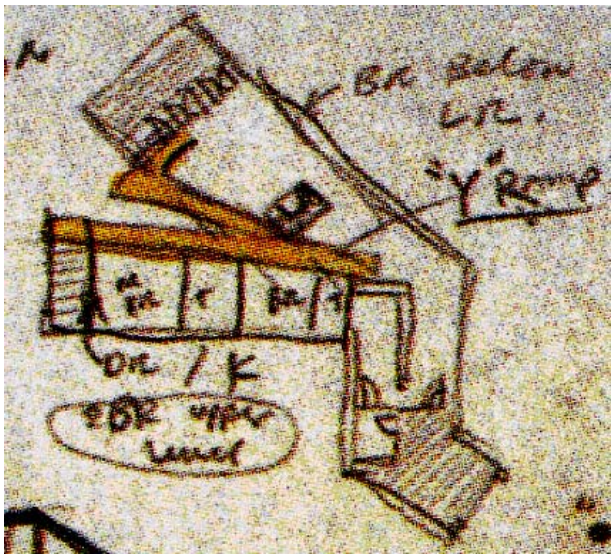


Figure 1.3 Text labels in the concept sketch for Y House indicate concerns about spatial arrangements of functional spaces. (Source: El Croquis, Holl, 2002)

Architects also consider circulation paths in their design. For example, in Figure 1.4, the drawing

has many graphical symbols such as lines and arrows. These lines represent wall partitions. The curvy arrows represent the circulation paths. This is evident because the architect wrote the text "Freedom of Movement" in the upper left of the drawing. The presence of the many lines demonstrates the designer's concern about people's movement through the space.

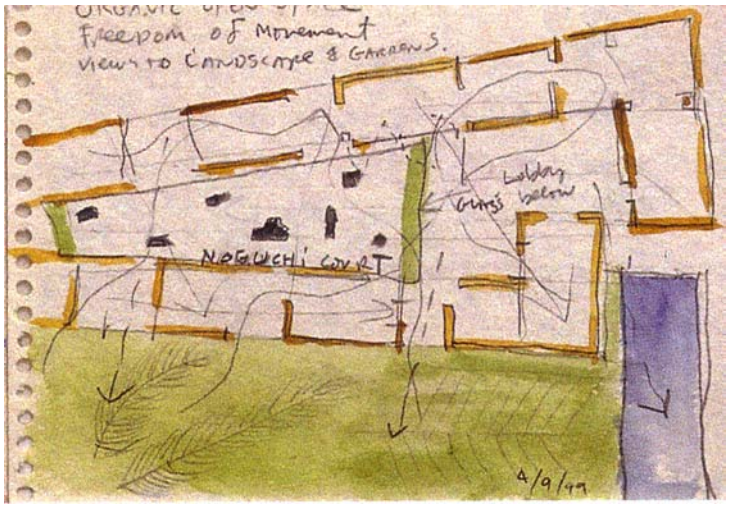


Figure1.4 Circulation path concerns represented as curvy arrows in Holl's concept sketch of the Nelson Atkin Museum of Art Expansion (Source: El Croquis, Holl, 2002)

1.2.3 3D Visualization

Architects use 3D perspective or isometric drawings during the design process to reason about form and functional arrangements. Often these plan and 3D drawings appear on the same piece of tracing paper or on pages in the same sketchbook. Figure 1.5 (a) shows a 3D drawing that appears right below the plan drawings of the Y House on the same page. This figure illustrates that the designer was concerned about the look and feel of the 3D form when he represented his design ideas in 2D drawings. Figure 1.5 (b) shows a bird's eye view of the plan sketch (left). One can see that the relations among rooms are clearly illustrated in this drawing by simply extruding the wall lines from the plan diagram. The circulation path here is also colored in yellow like the plan

diagrams (Figure 1.5 (b) or Figure 1.1).

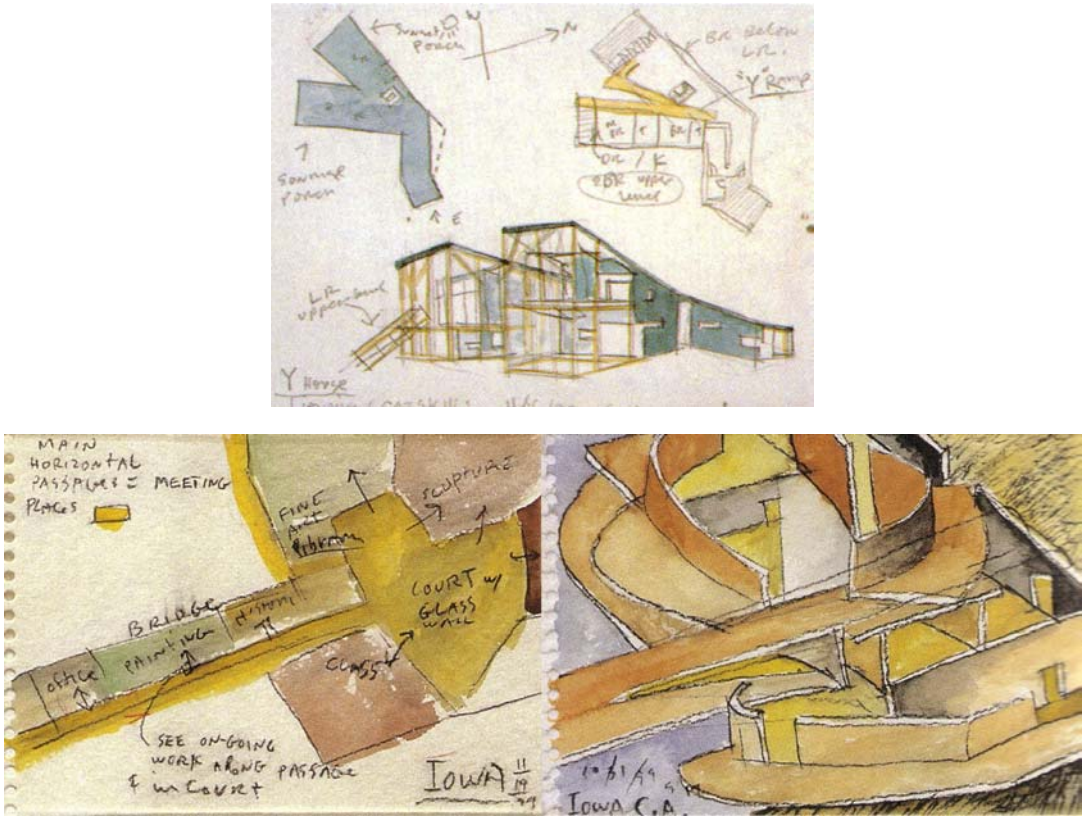


Figure 1.5 (a) Concept Sketch, Y House: (b) Concept Sketch, University of Iowa's Art and Art History Building (Source: El Croquis, Holl, 2002)

In two previous sections, we have seen two things examined (section 1.1). First, the designers make rules and use the rules in designing process. These rules are helpful to reduce the errors. Second, the reasoning issues (section 1.2) are deciphered through the Steven Holl's sketches such as spatial concerns, functional concerns and 3D visualization. This thesis proposes a sketch-based system with critiquing for avoiding errors. The system is based on the idea that various design concerns are expressed graphically on drawing and can be interpreted to provide design critiques.

1.3 GOALS OF THE THESIS

This thesis has three goals. The first goal is to integrate feedback into a sketch design system. Most CAD systems enable the designer to create external design representations (i.e. drawings) but they do not provide feedback. Feedback during the design process can appear as a form of critiquing. Critiques are critical reviews or evaluation results about the designer's artifacts. Critiquing provides the designer with knowledge that may help to improve the design. Most critiquing systems and checker systems (e.g., code checking and floor plan checking) have their own built-in structured drawing editors for interpreting the designs. Most critiquing system researchers invented custom drawing editors that work only with their critiquing system. These editors make it easy to capture the necessary knowledge from design proposals and to generate the design critiques related to the design proposals (Hu et al. 2000). However, these editors are not in widespread use. For example, if an architect designs a hospital and then wants to receive knowledge-based critiques about his/her design, the critiquing system cannot interpret the drawings. S/he must re-enter the design drawings using the drawing editor of a critiquing system. Moreover, these drawing editors are often so structured that their use as an early design tool is limited. These structured editors impose an additional cognitive load to the designer, because the designer must follow the editor's structure (e.g. menu, coordinate input method, etc.). Most designers begin design by sketching and diagramming and only later move to structured CAD editor to draft the final result or construction drawing. Many design decisions are made by the time the design moves from sketch to CAD. Therefore, it would be useful to apply the critiquing system as early in the design process as possible, ideally in the sketch stage when the designer' ideas are still fluid and open to change. If the editor is sketch-based, it will be much better and easier to explore the design ideas. Therefore, we are interested in incorporating critiquing into sketch-based early-stage design systems.

Designers usually prefer to sketch with pen and paper in the early stages of design. A sketch-based computer-aided design system is a tool to support the free exploration of ideas. Sketching allows designers to interact with their sketches. With sketching designers create, examine, and explore ideas. We are interested in building a computer based design tool with which, as with traditional tools, a designer can examine and generate design solutions stimulated by mental imagery from sketches (Fish and Scrivener 1990). Moreover, such a system would allow designers to improve or revise their designs with the provided critiques.

The second goal of the thesis is to investigate what kind of design constraints can be translated to rules. In the previous section (Section 1.2), we noted how one architect, Steven Holl, recorded his reasoning of design issues about spatial, functional relation and 3D visualizations in drawings. He uses visual symbols and shorthand notes to record his design ideas and concerns. Holl was reasoning about the relation of neighboring rooms as well as the whole arrangement with dividing larger functional spaces. Figure 1.3 and 1.4 show his concerns about functional relationship and circulation path. Using his drawing, he reasons about horizontal or vertical adjacency. He also uses 3D perspective drawings to visualize the form and relationship of spaces. To support architects' designing activities about spatial and functional relationships and 3D spaces in their design drawings, we built Design Evaluator as a proof-of-concept system to demonstrate how a computer program can support the designer's activities with provided critiques.

The third goal is to investigate how to visualize the design knowledge to the designers. The important component of the Design Evaluator is critiquing. The important thing is how to display critiques which encapsulated knowledge to the designer. The design knowledge in Design Evaluator is visualized in three ways; textual, graphical and three dimensional display. The Design Evaluator connects these three visualization methods with each other.

1.4 OUTLINE OF THE THESIS

This thesis is organized as follows. Chapter 2 describes a scenario using Design Evaluator. Chapter 3 describes the concepts and fundamental ideas of the Design Evaluator system. Chapter 4 describes the implementation of the Design Evaluator system. It explains the design environment, sketch interface, checkers, and the critique display methods. Chapter 5 describes the roles of design drawings and related work on sketch design systems and critiquing systems. Chapter 6 concludes with the summary of the thesis and discusses future research directions.

Chapter 2

A Hospital Design Scenario

2.1 HOSPITAL SCENARIO

This chapter describes a scenario of how a designer might use Design Evaluator in her design process. The scenario presented in this section motivated the implementation of Design Evaluator. All illustrations in this chapter are actual screen shots of the functional Design Evaluator system. The following sections describe a use scenario of the Design Evaluator that attempts to support the design concerns as outlined in the previous chapter.

We use the Design Evaluator system for a hospital design scenario. A hospital is a very complicated environment where the patients are cured, where the medical staff is educated, and where all the sorts of activities take place. Hospital designs are so complex because many spatial and functional issues have to be dealt with simultaneously. It is also a life-and-death matter when errors would be costly to fix.

In the early stages of design, an architect usually decides the program analysis with zone organizations. Hospital designs typically have three zones: Clinical, Nursing and Support. First, the architect starts to research what spaces are needed in the hospital. She then decides the placements of rooms in each zone through program analysis. For example, the ER (Emergency room) and ICU (Intensive Care Unit) should be in the clinical zone. Although these placement decisions seem simple and obvious, in a complicated building like a hospital, it is not uncommon to find incorrectly placed rooms from a post-occupancy study (Kobus, 2000).

After initial arrangement of spaces, the designer starts to study the spatial requirements in the

hospital design, such as adjacency requirements, proper room sequence requirements and minimum room sizes. At this stage the designer would examine circulation path sequence through the different rooms for different stake holders and activities. For example, if she considers the path (…- Hall - OR [Operation Room] - PACU [Post Anesthesia Care Unit] - Preoperative area - ICU - Ward …), she will realize that this path has an improper sequence of rooms: OR - PACU- Preoperative area. This path is improper because the path should follow the surgical process which happens in the following order: Preoperative area - OR - PACU.

She then studies the adjacency requirements for different operation. For instance, if she considers the path (ICU [Intensive Care Unit] - Nursing Station - Hallway - Ward - Inpatient Surgery - ER [Emergency Room]), this path violates the adjacent requirement for ICU and ER. The ICU and ER should be adjacent for effective medical treatment and patient delivery (Kobus 2000). The path in the example described above is not proper, because the ICU and ER are too far apart.

Finally, the designer checks the room dimension against the fundamental requirements such as inpatient capacity and the sizes of necessary appliances and furniture. If the room is smaller than the requirement, the room cannot carry out the function and should be revised accordingly. Below we describe a scenario of designing with the Design Evaluator.

2.2 A DESIGN SESSION WITH THE DESIGN EVALUATOR

Karen, an architect, is using the Design Evaluator to design a hospital. Design Evaluator acts as an intelligent assistant that has the required design knowledge to alert Karen if there is any possible problem or concern regarding her spatial arrangement design. Figure 2.1 shows the environment of the Design Evaluator. The Design Evaluator environment consists of a Sketch Window, a Tool Palette, a Verbal Critiques Window, and a 3D Visualization Window. Karen starts

by sketching a design using a stylus and a digitizing tablet on the sketch window. Karen will also receive critiquing message as visual feedback in the sketch window (described in section 2.5 and 2.6)

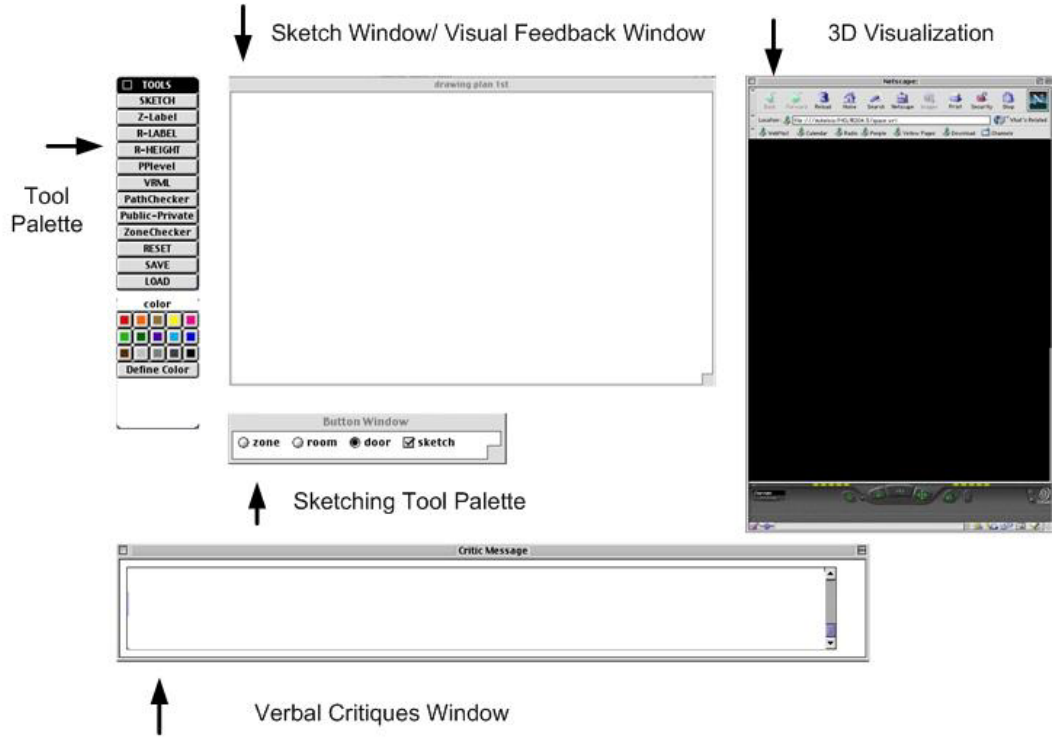


Figure 2.1 Design Evaluator Environment: Sketch window/Visual Feedback Window, Tool Palette, Verbal Critiques Window, and 3D Visualization Window

2.3 DIAGRAMMING WITH SKETCH AND LABEL OF ZONES, ROOMS AND DOORS

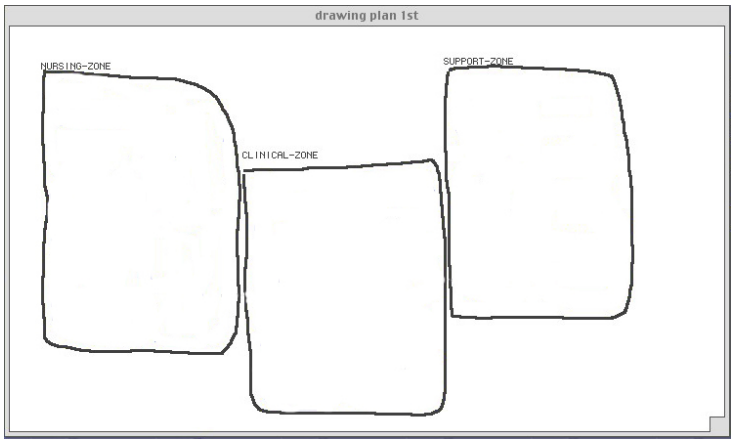


Figure 2.2 Karen draws three zones of Nursing zone, Clinical zone and Support zone in the Sketch Window

Karen begins by sketching three different zones: Nursing-zone, Clinical-zone, and Support-zone (Figure 2.2) and then she labels each zone. She sketches rooms in each zone, for example, Karen draws six small bubbles in the NURSING-ZONE and then named them, WARD, VISITORS-ROOM, ICU (Intensive care unit), Staff (Medical staff station), Nursing (Nursing station) and HALLWAY. Karen continues drawing other rooms in the other zones (Figure 2.3). She draws lines to connect the space bubbles and these lines represent the doorways.

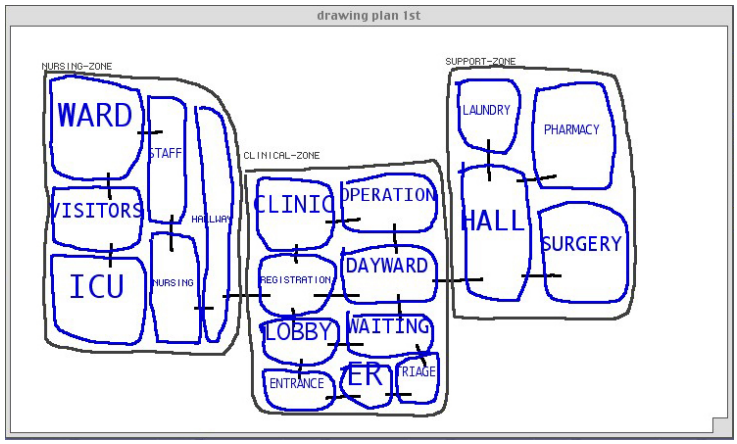


Figure 2.3 Sketch Diagrams: sketch zones, rooms, and doors

Design Evaluator observes all Karen's actions, records all the room and zone labels, and builds a database of all this information. From this database, Design Evaluator understands which zone contains what rooms, which room is connected to which room through which doorway, and so on. This will be used later to generate design critiques.

2.4 RECTIFY THE DRAWINGS

Karen has to present the preliminary design to the board of trustees of the hospital in the afternoon. She thinks that a hard-line drawing is better for the presentation than her freehand sketches, so she unchecks a "sketch" check-box in the Sketch Tool Palette. She instantly receives a rectified drawing from her freehand sketch and ready for the presentation (Figure 2.4).

Design Evaluator can display the designer's floor plan drawing in two modes: sketch mode and rectified mode. Figure 2.4 shows the rectified mode where Design Evaluator turns the bubbles in the designer's sketch diagrams into rectangles. Every element in the floor plan can be represented in the sketch mode as well as the rectified mode. In the sketch mode, rooms and zones are displayed as the bubbles as exactly how the designer drew. Doors are displayed as the connecting lines. In rectified mode, zones and rooms are displayed as the beautified room rectangles. When Karen draws a bubble, the system computes the boundary of the drawn bubble and displays a bounding box instead. Doors are displayed as white space between two rooms.

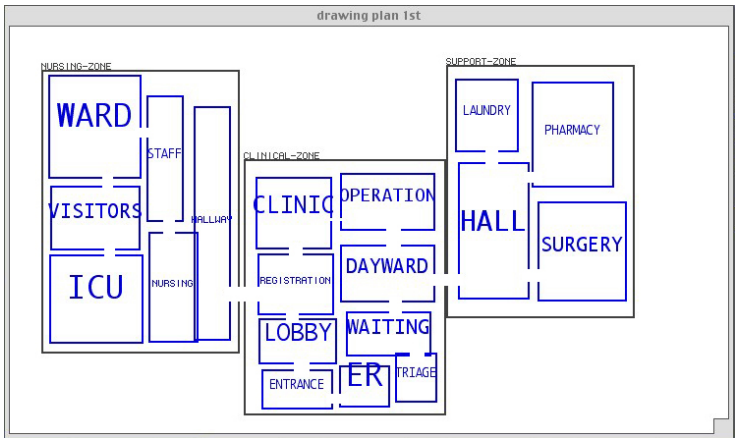


Figure 2.4 Rectified Sketch Diagrams

2.5 PATH CHECK

After Karen sketches the floor plan diagram, the system interprets the sketch and finds all the possible paths in the floor plan, that is, all the possible ways that a person could move through the building. The Path Checker finds the relevant paths and compared them with predefined rules. For example, if one rule says the (ICU(Intensive care unit) AND ER SHOULD BE ADJACENT), the system finds the path that has the ICU and the ER and checks whether these two rooms are adjacent.

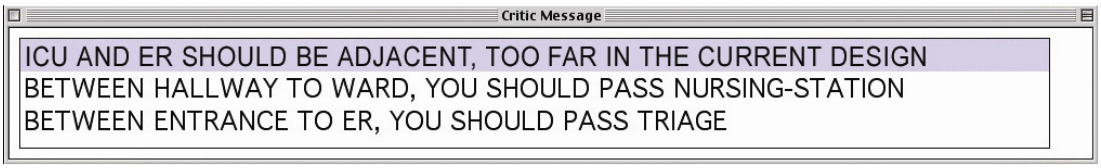


Figure 2.5 Verbal Critiques Messages in the Verbal Critique Window (Path Checker)

The Design Evaluator sees that Karen's design might cause problem, so it points out the problem and suggests a fix. For example it displays "BETWEEN ENTRANCE TO ER, YOU SHOULD PASS TRIAGE" (Figure 2.5). The Design Evaluator reminds Karen that the path from the

entrance to the ER must pass through the TRIAGE area. The placements of functional spaces of ENTRANCE, TRIAGE and ER should follow a particular sequence of ENTRANCE – TRIAGE – ER. In other words, to access the ER, the circulation must pass through TRIAGE.

Karen sees that there are 3 messages displayed on the screen. She clicks on the first one that states “ICU AND ER SHOULD BE ADJACENT, TOO FAR IN THE CURRENT DESIGN” (Figure 2.5). The Design Evaluator draws a path on the sketch window from ICU and ER to indicate the path in question (Figure 2.6 top). The connection between verbal and visual critiques helps Karen to identify possible design errors.

Then, she clicks the second line, which states “BETWEEN HALLWAY TO WARD, YOU SHOULD PASS NURSING-STATION” and she sees the path from the hallway to the ward displayed in Figure 2.6 (bottom).

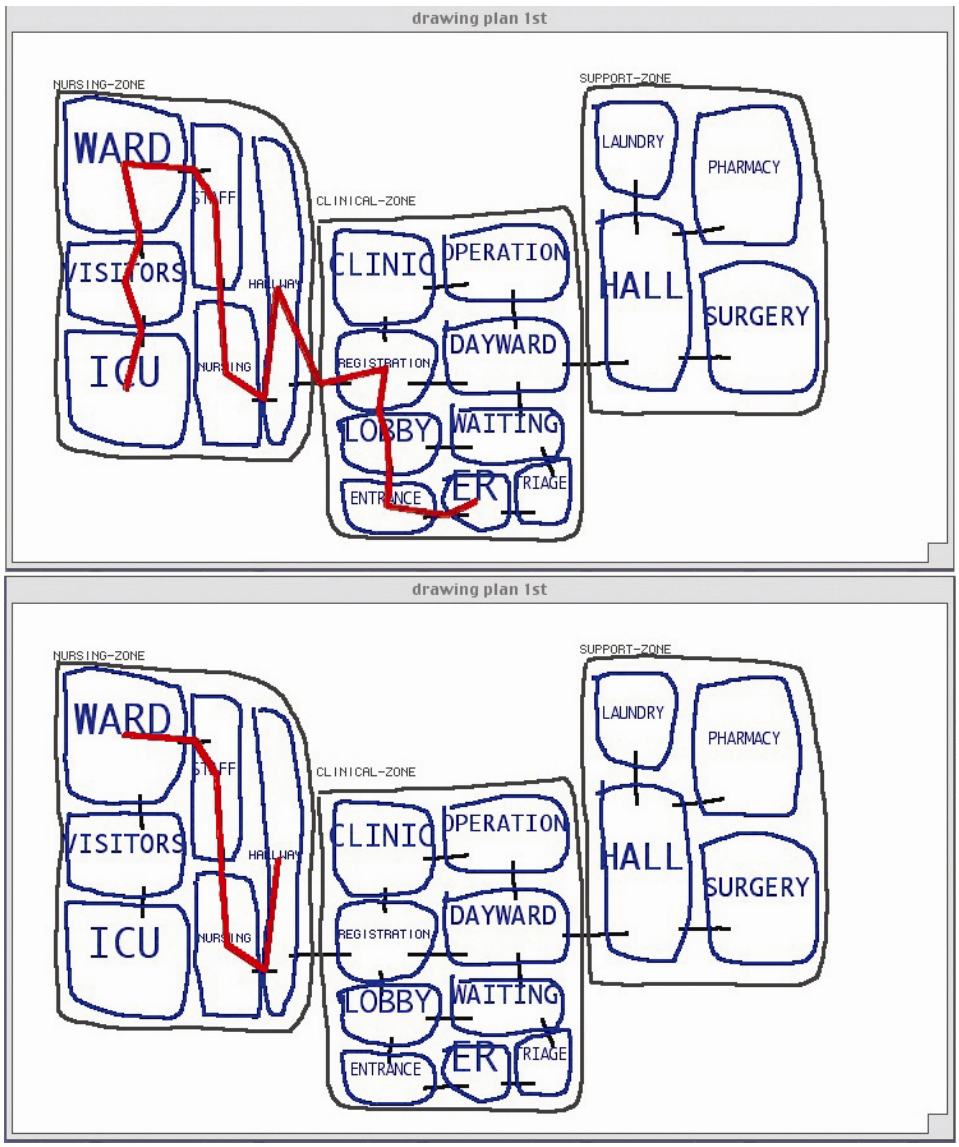


Figure 2.6 Path Checker: Design Evaluator displays the problematic path as thick red lines when Karen clicks the text message in Figure 2.5 (top: ICU to ER, bottom: HALLWAY to WARD)

2.6 ZONE CHECK

After Karen sketches the floor plan diagram, another kind of diagnostic message appears in the text critique window. It tells Karen that "ICU SHOULD BE PLACED IN CLINICAL-ZONE" and the "INPATIENT-SURGERY SHOULD BE PLACED IN CLINICAL-ZONE" (Figure 2.7). The Design Evaluator knows many hospital planning rules such as room placement requirements.

For example, the ICU should be placed in the clinical zone, because this room has the character of clinical functions and needs adjacency for easy patient transportation to other clinical rooms.

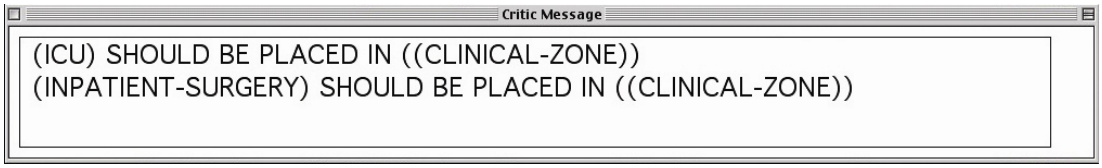


Figure 2.7 Verbal Critiques Messages in the Verbal Critique Window (Zone Checker)

When Karen clicks on this message, the sketch window highlights the improperly placed rooms (ICU and INPATIENT-SURGERY) with the thick red lines (Figure 2.8).

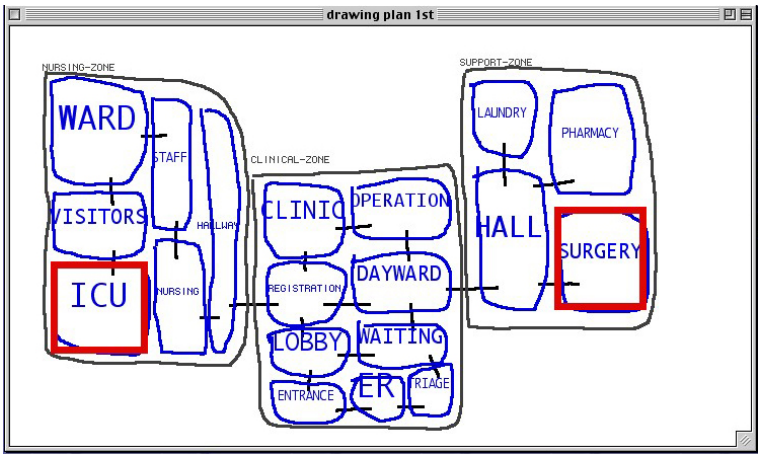


Figure 2.8 Zone Checker: incorrectly placed rooms are highlighted in red thick lines (ICU and SURGERY)

Responding to the critiques that are displayed verbally and visually, Karen moves the offending rooms into the proper zone as suggested by the text critiques. She moves the ICU into the Clinical Zone.

2.7 MODIFICATION OF FLOOR PLAN

Karen continues to revise and design the hospital space layout with the help from the Design Evaluator's verbal and visual critiques. For example, she erases two rooms to make space for moving the ICU into the clinical zone. She then moves the ICU over to two rooms' former location (Figure 2.9).

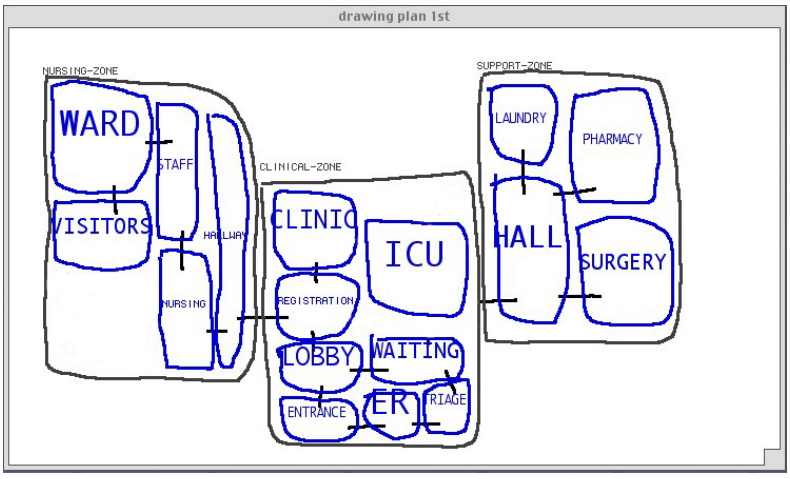


Figure 2.9 Karen moves the ICU from Nursing-Zone to Clinical-Zone

2.8 AREA CHECK

Once Karen decides the capacity and the functions of the rooms in the schematic design stage, she asks the Design Evaluator to do an area check. The area check indicates to Karen that "ER IS TOO SMALL, THE MINIMUM AREA IS 7000" (Figure 2.10) to alert her about the adequate size for this space. This message appears because the Design Evaluator has a list of the minimum size requirements for all functional spaces. For example, the dimension of the ward should be decided by the inpatient capacity number of the hospital and the sizes of medical supplies or

furniture. All these minimum area requirements are recorded as rules and when any of the rules are violated, the message will appear in the critiquing window. The Design Evaluator calculates the area of each space, uses the labels to find the minimum area requirement for that space, and prints the message if that requirement is not met.

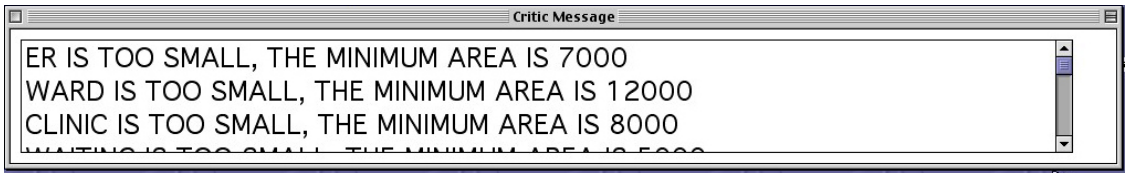


Figure 2.10 Verbal Critiques Messages in the Verbal Critique Window (Area Checker)

2.9 3D VIEW

To get a better view of the design problem in 3D space, and to see the problem from the hospital user's perspective Karen now takes a look at the walk-through model that the Design Evaluator has generated from the sketch (Figure 2.11). These models are texture-mapped with photographs taken from rooms of different functional spaces in a hospital. For example, the emergency room in Karen's design is texture mapped with photographs of a real ER. Texture-mapped models give Karen a realistic and convincing simulation of the designed space. This makes it easy for her and the board of trustees to visualize the spatial relations in 3D and be able to "walk" inside the simulated space to further evaluate the spatial quality of her design.

Figure 2.11 3D Texture-mapped 3D VRML Models: Operating Room, Ward, Nursing Station, Hallway and Physical Therapy Room

As shown in Figure 2.11, the Hallway space is texture-mapped with photos of a hallway. In the 3D view Karen sees the HALL photos proceeds to “walk” into the WARD to examine it. Moving the viewing angle, Karen sees that the next room is a patients-ward (WARD) and realizes the placement of HALLWAY next to WARD should be reconsidered.

Karen then notices that NURSING-STATION should be placed next to WARD for effective medical treatment flow (Figure 2.12). This 3D visualization helps Karen to detect problematic room arrangements and helps her to locate herself in the 3D visualization of the floor plan she has designed.

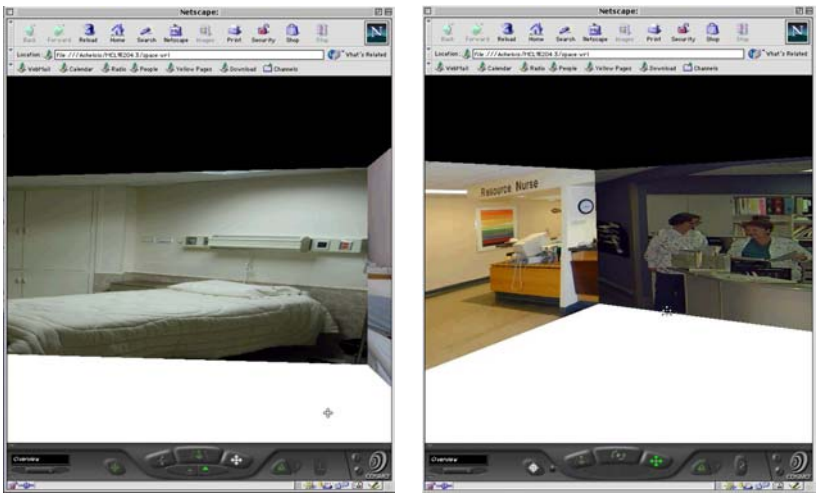


Figure 2.12 Texture-mapped Space: WARD and NURSING-STATION

2.10 SUMMARY

The above scenario is a working demo of the Design Evaluator system. We can see from this scenario that Design Evaluator system supports designer’s floor plan diagram sketching in several ways: (1) Design Evaluator recognizes all the elements such as zones, rooms and doors. (2) Design Evaluator interprets the floor plan diagram and produced lists of all possible paths, (3)

system checks the design diagram with the predefined zone rules and path rules to offer design critiques. The Design Evaluator displays the critiques in three ways: verbal critique messages, visual feedback on the floor plan diagram and a texture-mapped 3D visualization.

Chapter 3

Design of the Design Evaluator

This chapter describes the overall design of the Design Evaluator system. The following sections explain the design environment and what each component performs for supporting architect's design.

Table 3.1 illustrates how the designer works with the support of the Design Evaluator system. The Design Evaluator supports the designer's activities in each step; recording the drawing, comparing the design ideas with the predefined rules, and visualizing the critiques in three ways. These activities of the Design Evaluator support the designer's designing.

Table 3.1 Activities of the Designer and Design Evaluator in the Design Process

	Designer	**Design Evaluator**
Description	Draw	Record drawing
Evaluation	Reflect	Compare design with requirement rules
Visualization	Imagine, See, Read	Text, Plan annotation & 3D view

We can see the design environment as consisting of three layers (Table 3.1); ***description layer***, ***evaluation layer*** and ***display layer***. In the ***description layer***, the designer proposes a floor plan diagram by drawing. Then the designer reasons about the proposal with design criteria in the ***evaluation layer***. The design criteria in the current version of the Design Evaluator include three kinds of rules; (1) path rules, (2) zone rules and (3) area rules. This layer checks the design

proposal with predefined rules. The Design Evaluator system displays the result of evaluation as critique messages in the ***display layer***.

3.1 DESIGN ENVIRONMENT

Table 3.2 illustrates the character and activities of each layer. In the following part, we describe these three layers.

Table 3.2 Three Layers of the Design Evaluator: Description Layer, Evaluation Layer and Display Layer

Layers	Character	Activities
Description Layer	[Input]	Sketching and Labeling
Evaluation Layer	[Rule Check]	Comparing with Zone, Path and Area Rules
Display Layer	[Display]	Three Display Modes; Verbal Message, Visual Feedback, & 3D Visualization

3.1.1 Description Layer (Sketching + Labeling)

The Description Layer of the Design Evaluator enables the designer to describe the arrangement of a proposed design by creating a labeled sketch. Designers usually augment their sketches with shorthand notes. Most architects find that sketching is a good and easy way to quickly explore design ideas freely. The description layer of the Design Evaluator supports the designer to sketch her/his design ideas. The floor plan sketch contains the information that the Design Evaluator needs in order to retrieve relevant knowledge from a database of design criteria rules.

3.1.2 Evaluation Layer

The Evaluation Layer links the Description Layer with design criteria and conveys design feedback to the Display Layer. The Checker component (see 3.5 Section) of the Design Evaluator checks the design sketch against the design knowledge base of predefined rules. If the Evaluation Layer finds a rule violation, it generates several critiques and sends them to the Display Layer.

To give the designer appropriate feedback, the system must have embedded design knowledge. Design knowledge in this case is stored as rules. Some design requirements and concerns can be translated into rules. We put about 30 spatial and functional concerns in the Design Evaluator system. For example, one rule specifies that the ICU and the ER should be adjacent. An adjacency requirement like this is represented a rule that the Checker uses to compare with the relationships recorded in the Description Layer.

3.1.3 Display Layer

The Display Layer handles three modes of critiquing display: (1) verbal feedback of text advice, (2) visual feedback as annotated drawings, and (3) texture-mapped 3D models. These displayed critiques describe why a particular part of the current design has a potential problem and which design constraints and requirements are violated.

3.2 COMPONENTS OF DESIGN EVALUATOR

3.2.1 System Overview

Design Evaluator has four components: a sketch interface, a database, three checkers and a display manager. Figure 3.1 shows the relations of these four components and the information flows between them. The *Sketch System* and the *Design Database* is included in the Description Layer, the *Checkers* is in the Evaluation Layer, and the *Display Manager* is contained in the

Display Layer.

The *Sketch System* supports sketching of spatial diagrams that consist of visual elements and identifies the bubble diagram and text labels. It interprets the spatial relations from the diagram and generates all the circulation paths. There are two kinds of *database* in the Design Evaluator; (1) the record of all sketched objects and the generated circulation paths and (2) predefined rule base that specifies design criteria. Then the *Checkers* examine the sketched diagram, comparing it with the rule database. If the *Checkers* find violations in the design, the *Display Manager* displays the design critiques in three ways: text messages, annotated drawing and 3D visualization.

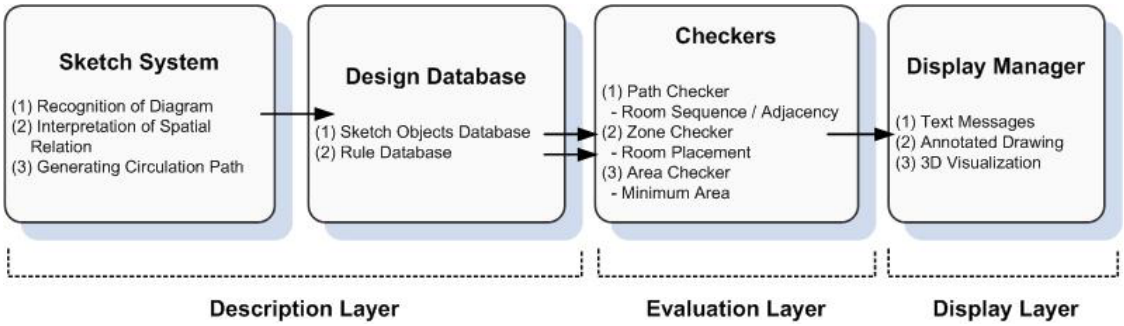


Figure 3.1 Information Flow between in Four Components of Design Evaluator

3.2.2 The Design Evaluator Environment

If we look at the Design Evaluator from a user's perspective, the user interface is composed of three windows (sketch window, verbal critiques window, and 3D visualization window) and two tool palettes (Figure 2.1) (tool palette and sketch tool palette). The user can choose commands for sketching, labeling and checking in the toolbar. The sketch window has two functions. It displays freehand strokes drawn by the user with a digitizing tablet and pen. It also displays as graphical critiques in the form of annotated diagram on the floor plan. The 3D visualization window shows

the three dimensional model generated from the floor plan.

3.3 SKETCH SYSTEM

3.3.1 Recognizing Floor Plan Diagram Elements

In Design Evaluator freehand diagrams that represent spatial arrangement of rooms in a floor plan are drawn with a pen. Designers enter two types of data into their drawings: spatial diagrams and text labels. Spatial diagrams of drawn shapes are recognized by Design Evaluator system as functional zones and rooms with doorway connections. Design Evaluator recognizes two kinds of bubbles: zone, room, and connecting lines as doors. The designer uses a type-in box to input a text label for each room and zone.

3.3.2 Interpreting Spatial Relations

The *Sketch System* recognizes several spatial relations among the drawn elements: the relations of rooms and zones and the relations of rooms and doors. When a room is located in a certain zone, the *Sketch System* recognizes the relation between the room and the zone as “containing”.

3.3.3 Identifying Circulation Paths

Once all spatial relations are recognized and recorded, the system generates a list of all possible circulation paths moving from space to space. This is important because people experience the space through the circulation pattern within an architectural space.

3.3.4 Two Modes of Display

The sketch window has two modes of display: sketch mode and rectified mode. In sketch mode the designer draws bubble diagrams to represent functional space such as entrance and triage and

draws lines to connect bubbles to represent connections between functional spaces (Figure 3.2 - left). When the ‘rectified’ mode is selected, the system will display the same diagram in a ‘rectified’ mode. In this mode, a freehand bubble will be converted into a rectangle shaped room and doorways are shown as open gaps along the wall lines of the room (Figure 3.2 - right).

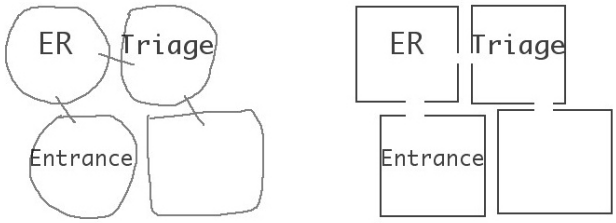


Figure 3.2 A Simple Floor plan in Sketch and Rectified Mode

Text labels of rooms inputted from a type-in box are resized automatically to fit inside the space. This is achieved by adjusting the font size of the text according to the width of the bubble. The font size is calculated by dividing the width of the bubble by the numbers of characters from the type-in. After the proper font size is derived, the system displays the text label in the center of bubble that represents a room. If the bubble represents a zone, the system displays the text label on the left corner of that top zone bubble.

Figure 3.3 illustrates the process of converting sketch bubbles into rectangles and labeling the drawing. Design Evaluator stores all the information of a sketch object (e.g. coordinates, point lists, bounding box, label, etc.). Figure 3.3 shows the process of how a certain room is rectified in the floor plan diagram. To render the sketch in rectified mode, the system calculates the bounding box (the maximum and minimum coordinates) of each room bubble (Figure 3.3-(a)) and then displays a rectangle (Figure 3.3-(b)). In the Figure 3.3-(a), the boundary lines are from the maximum and minimum coordinates and indicate the bounding box. The system then displays the

room's name in the middle of the rectangle (Figure 3.3-(c)).

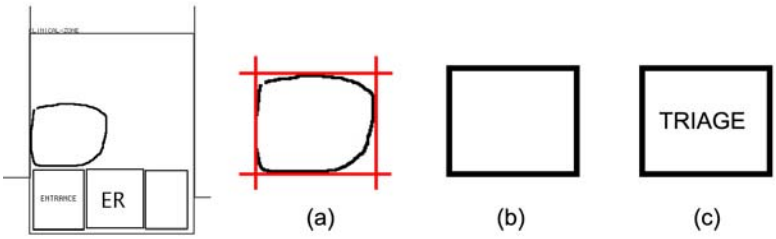


Figure 3.3 Drawing Process in the sketch interface. Left figure is the part of the sketch diagram. (a) sketch bubble diagram and recognizing the boundary of bubble; (b) rectified drawing after recognizing boundary of bubble diagram; (c) labeling the functional names of zones and rooms

3.4 DESIGN DATABASE

Design Evaluator is an object-oriented program. Its data structures include two kinds of information (1) a sketch object database and (2) a rule database. First, the system stores all the sketch objects (rooms, zone boundaries, and doors between rooms) and then generates a list of all possible circulation paths. Secondly, the system contains a database of design rules that are later used to generate design critiques. Figure 3.4 shows these object classes with their important slots. An object class is a kind of blueprint that defines the instances and variables common to all objects of a certain kind. For example, zone class defines that all zone object instances have a label, coordinates, and rooms as variables. In other words, an object is a kind of bundle of variables. Each room object contains five variables, these decides the character of object. However, each instance of zone object has different values stored in these variables.

Figure 3.4 illustrate the Sketch Object Database and rule database. The *Sketch System* stores all information about objects in the Sketch Objects Database. The Design Evaluator has three kinds of class. The Zone Class defines each zone object as having an id, label, coordinates and rooms.

Room Class defines that each room object has id, label, coordinates, doors, height. Finally, Door class defines that each door object has id, coordinates, room1, and room2.

Based on the stored information of these three kinds of objects, a list of all possible circulation paths is generated. This path list is a sequence of room and door objects and is stored in the Sketch Object Database. In the other hand, the Rule Database has three kinds of rules; Zone Rule, Path Rule, and Area Rule. Zone Rules relate to the *proper room placement.* Path Rules are of two kinds, related to *room sequence* and *adjacency*. Area Rules are related to *minimum area*.

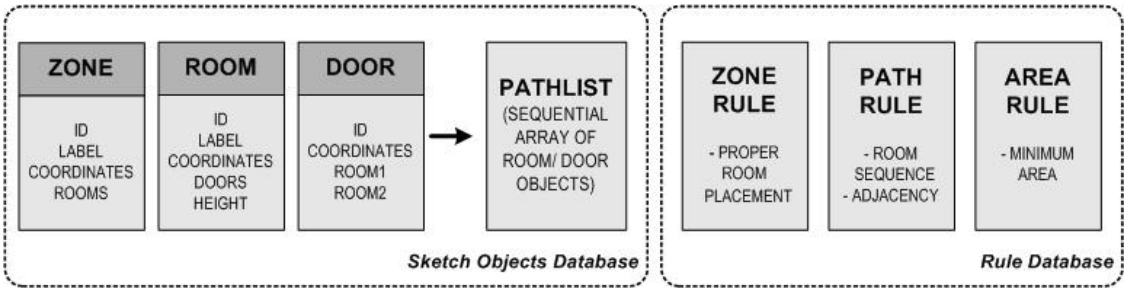


Figure 3.4 Sketch Objects Database and Rule Database

3.4.1 Database of Sketch Objects

Three main lists are the heart of the Design Evaluator's design representation; zone list, room list and door list. Each zone object stores a list of rooms that it contains. Each room stores a list of its doors. Each door object knows which rooms it connects. All data of these sketch objects are connected with each other. Figure 3.5 shows the relations of objects.

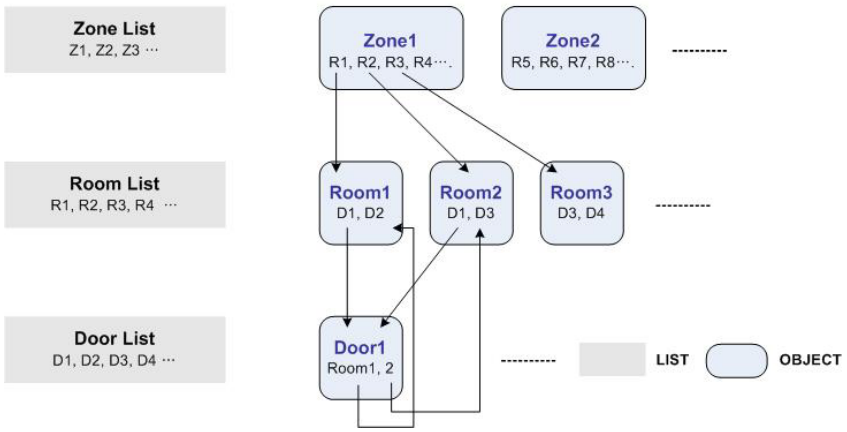


Figure 3.5 Relations of Sketched Objects; Each zone has a list of rooms and each room has a list of doors. Each door stores a list of the two rooms it connects.

Figure 3.6 illustrates how the system makes the interrelations with three kinds of objects. In Figure 3.6-(a), the designer has sketched one zone: CLINICAL-ZONE and three rooms: ENTRANCE, DAYWARD, and TRIAGE. These rooms are connected by three doors. From the sequence of drawing, we call the line connecting between DAYWARD and ENTRANCE as DOOR1, the line connecting between ENTRANCE and TRIAGE as DOOR2, and the line connecting between DAYWARD and TRIAGE as DOOR3. Figure 3.6-(b) shows the main lists that the Design Evaluator has constructed as the designer sketches in the Figure 3.6-(a) diagram. Figure 3.6-(c) illustrates how the sketched objects interrelate with each other in the database.

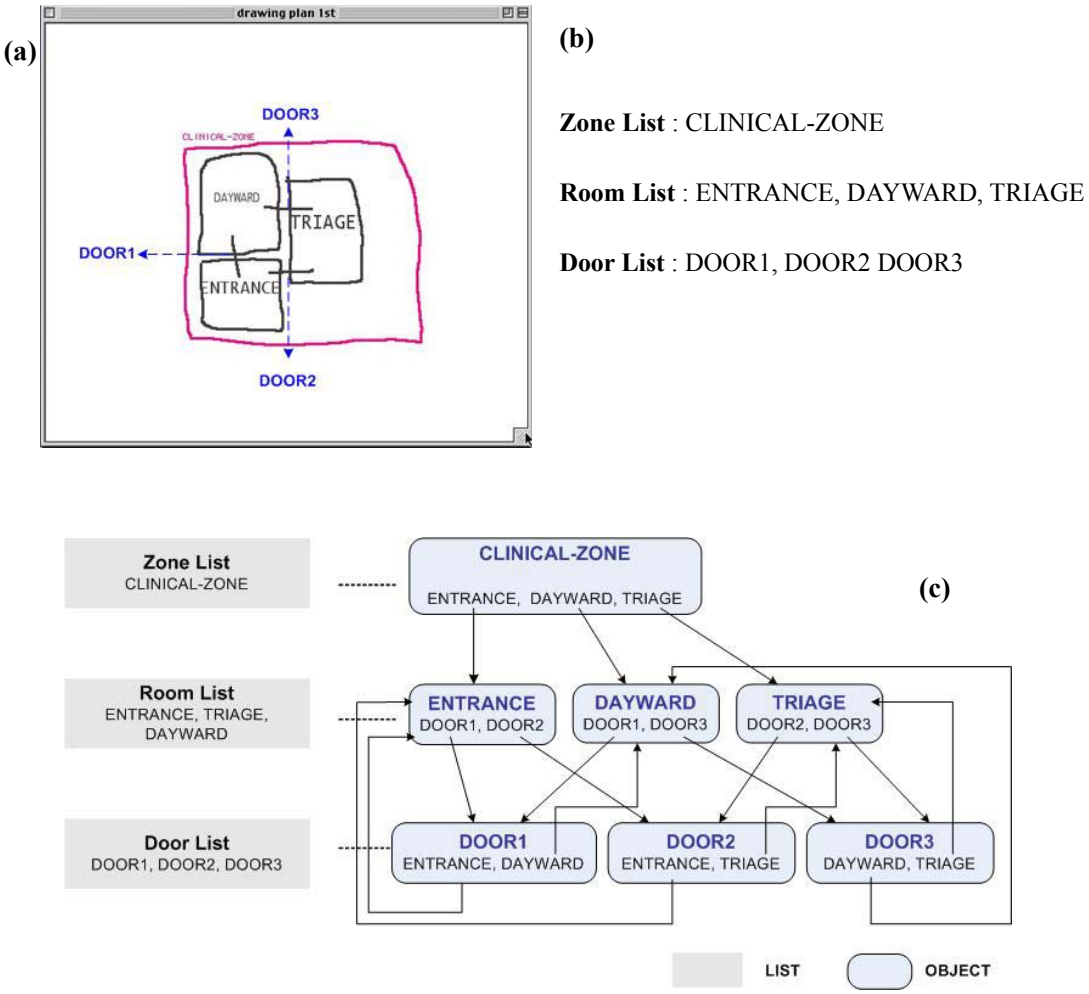


Figure 3.6 Simple Example of the Interrelated Sketched Objects; (a) Sketched Diagram (b) Three Main Lists: Zone List, Room List and Door List (c) Diagram shows how the sketched objects connects with each other

3.4.2 Database of Built-in Rules

To offer design critiques, the system needs domain knowledge. The design criteria are categorized as Zone Rules, Path Rules and Area Rules. These predefined rules are stored as symbolic text expressions.

The rules are previously defined by the designer to the Design Evaluator system. For example, the architect can input rules for proper room placement, adjacency, circulation sequence, and

minimum area. Path Rules includes two kinds of rules: 1) path sequence rules, and 2) room adjacency rules. The Zone Rules currently only deals with room placements in the appropriate zones. The Area Rules are about the minimum size of the rooms. In summary, there are two kinds of Path Rules, one kind of Zone Rule and one kind of Area Rule. Below we briefly explain the expressions of these four rules.

(1) Room Sequence Rule (Path Rule)

The Path Checker takes the form of an expression of:

(<Required-Sequence> <room1> <room2> [<room3>]*)

This expression indicates that path sequence should follow room1– room2 –room3. For example, the following expression represents a required circulation sequence in a hospital design:

(MUST-PASS-THROUGH ENTRANCE TRIAGE ER)

This expression indicates the desired path sequence to be ENTRANCE, TRIAGE, and ER. In order to access ER, the circulation path must pass through ENTRANCE and TRIAGE.

(2) Adjacency Requirement Rule (Path Rule)

The Path Checker takes the form of an expression of:

(<Adjacency> <room1> <room2>)

An adjacency requirement indicates that two rooms must be adjacent. “Adjacency” means two

---

* [<room3>] indicates that more rooms can be added as option.

rooms should be placed next to one another. For example, the following expression represents a required adjacency of two rooms in a hospital design:

```
(SHOULD-BE-ADJACENT ER ICU)
```

This expression represents the rule that ER and ICU should be adjacent.

(3) Proper Room Placements Rule (Zone Rule)

The zone checker takes the form of an expression of:

```
(<Placement>  <Zone>  (<Room> <Room> <Room> <Room> ……))
```

This expression indicates that all the rooms in the list inside the inner parenthesis should be in the given Zone. For example, the following expression represents a typical room placement requirement in hospital design that states ER, TRIAGE, CLINICAL-FOR-OUTPATIENT, and DAYWARD, etc. should be placed in the CLINICAL-ZONE.

```
(MUST-BE-IN CLINICAL-ZONE (ER TRIAGE CLINIC-FOR-OUTPATIENT DAYWARD…))
```

(4) Minimum Area Requirement Rule (Area Rule)

The area checker takes the form of an expression of:

```
(<Minimum-area>  <room1>  <minimum-size>)
```

The expression indicates that room1 should be bigger than the <minimum-size>. For example, the following expression represents the minimum area requirement about the specific room which

states that the ER should be not smaller than 4000 square feet.

```
(MINIMUM-AREA ER 4000)
```

3.5 RULE CHECKERS - CRITIQUING

The Rule Checkers perform two functions: (1) compare the sketched floor plan diagram with rules, and (2) generate design critiques. Checkers compare the recognized spatial information with each rule. If a Checker finds a rule violation, it will generate design critiques to be displayed by the Display Manager (described in the next section). Figure 3.7 shows an overview of the critiquing process.

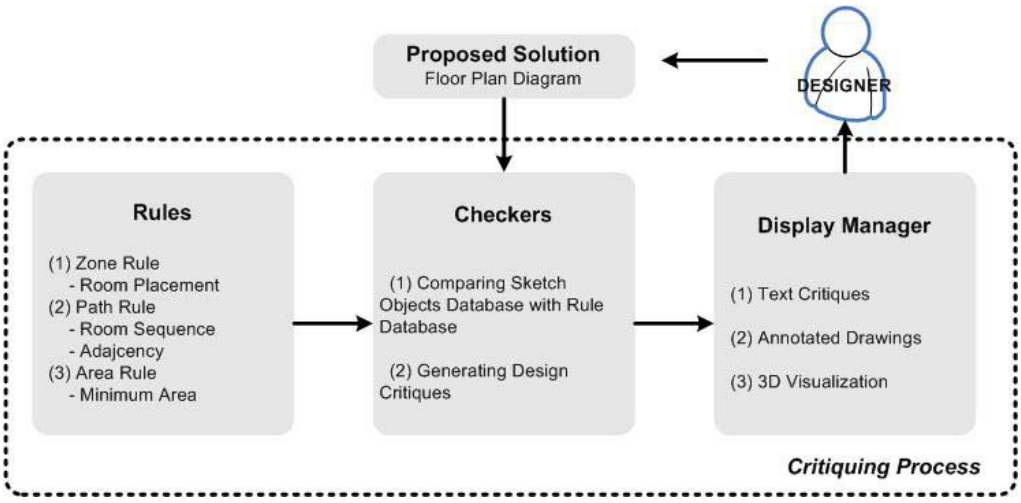


Figure 3.7 Overview of Critiquing Process. (a) User proposed design solution with sketched floor plan; (b) Analyzing the solution; (c) Comparing the solution with rules; (d) Generating design critiques, if there are errors or conflicts

Rule Checkers includes Zone Checker, Path Checker and Area Checker. First, the Zone Checker identifies any improper room placement in a zone. The Design Database contains all the spatial

information such as room placement, door locations and paths. It calls each zone rule and applies the rule statement as a true-false test to the relevant feature of current design. It then generates critiques for each improperly placed room it identifies and recommends the proper zone. Secondly, the Path Checker identifies potential problems with paths: 1) the sequence of rooms in the path and 2) the adjacency requirement. Third, the Area Checker checks whether the specific room has enough space to perform its particular function. It calls each area rule and compares calculated the room area with the minimum area prescribed by the rule.

3.6 DISPLAY MANAGER

Design Evaluator uses three methods to display the generated critiques: 1) text critiques, 2) annotated drawing, and 3) texture-mapped 3D visualization. This section describes the three ways the Design Evaluator system gives critiquing feedback.

3.6.1 Text Critiques – Verbal Feedback

Text messages are generated in a special critique window, when the checkers find problems in the proposed design diagram. The Design Evaluator has four functions: Room Sequence function, Adjacency function, Proper Room Placement function and Minimum Area function generated by the Checkers. Each checker generates its own critiques in a critique window. Figure 3.8 shows the critique message generated by the Path Checker. There are two kinds of rules in the Path Checker – room sequence and adjacency. Therefore, there are two types of critiquing message. The first type of critiquing message takes the room sequence function to produce the messages in the form of “BETWEEN HALLWAY TO WARD, YOU SHOULD PASS NURSING-STATION”. For example, the second and third lines in Figure 3.8 are generated by the room sequence function.

On the other hand, the first line indicates another critique: “ICU AND ER SHOULD BE ADJACENT, TOO FAR IN THE CURRENT DESIGN”. In the following section we describe how each checker function generates the text messages.

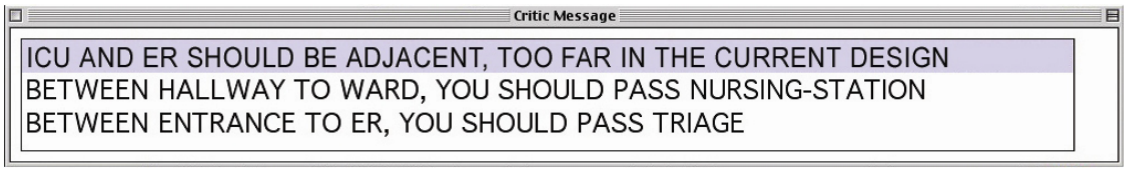


Figure 3.8 Text Critiques: Path Checker critique messages display adjacency requirement (1st message) and room sequence requirement (2nd and 3rd messages)

(1) Room Sequence function

“BETWEEN *Room1* **TO** *Room3*, **YOU SHOULD PASS** *Room2*”

If the path does not comply with the room sequence rules, this function inputs the variables (room1, room2, and room3) with the rooms that are placed in the different sequence from the rule.

(2) Adjacency function

“*Room1* **AND** *Room2* **SHOULD BE ADJACENT, TOO FAR IN THE CURRENT DESIGN”**

If two rooms that should be adjacent in the rule are placed too far apart, this function returns the critiques inputting the variables with the rooms.

Figure 3.9 shows the critique message generated by the Zone Checker. There is one currently only function in the Zone Checker – Proper Room Placement function. Therefore, there is one type of Zone Rule critiquing message. This function takes the proper room placement rule and then generates the critiques like the text critiques “ICU SHOULD BE PLACED IN CLINICAL-

ZONE" in Figure 3.9.

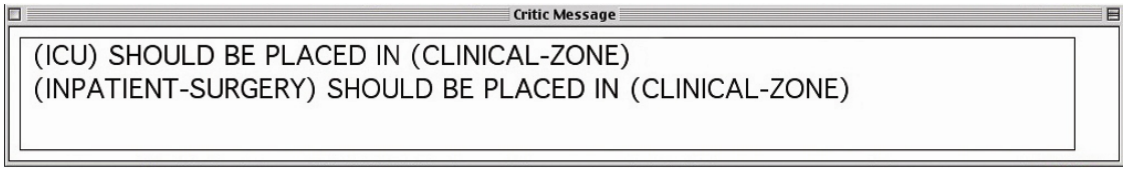


Figure 3.9 Text Critiques: Zone Checker critique messages signal problems with room placement.

(3) Proper Room Placement function

"*Room1* **SHOULD BE PLACED IN** *Zone1*"

After comparing the room arrangement with the zone rules, this function generates the critiques that indicate the desired zone placement. It fills two variables with the wrongly placed room and the desirable zone. The desirable zone can be found from the zone rules.

Figure 3.10 shows the critique message generated by the Area Checker. There is one kind of rule, minimum area requirement rule and it generates one type of critiquing message. The minimum area function takes the minimum area requirement rules, then generates the text critiques like "ER IS TOO SMALL, THE MINIMUM AREA IS 7000" in the Figure 3.10.

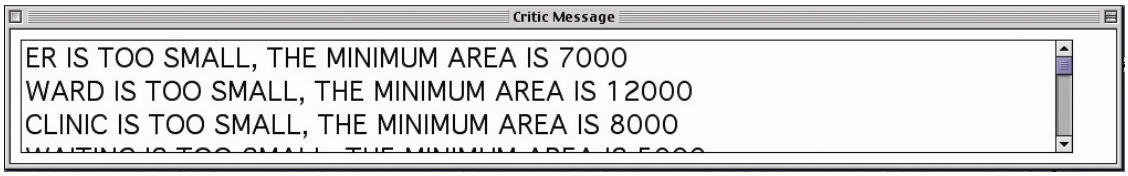


Figure 3.10 Text Critiques: Area Checker critique messages display the rooms that are not large enough to carry out the functions of rooms

(4) Minimum Area function

"*Room1* **IS TOO SMALL, THE MINIMUM AREA IS** *Minimum-Size*"

After comparing the area of the sketched room1 with the minimum-size of rule, the function generated the text messages, if the room1 is smaller than the minimum size.

3.6.2 Annotated Drawing – Visual Critiques

Each generated text critique message is connected with a drawing annotation. The Checker functions display the problematic paths and rooms as well as the text critiques. Therefore, the system can display the problematic paths and rooms as annotations on the drawing. These annotations are a kind of comment added to the design drawing. For example, when a problem space is identified, the system will highlight the boundary of that room with thick wall lines (ICU and SURGERY in red color in Figure 3.11).

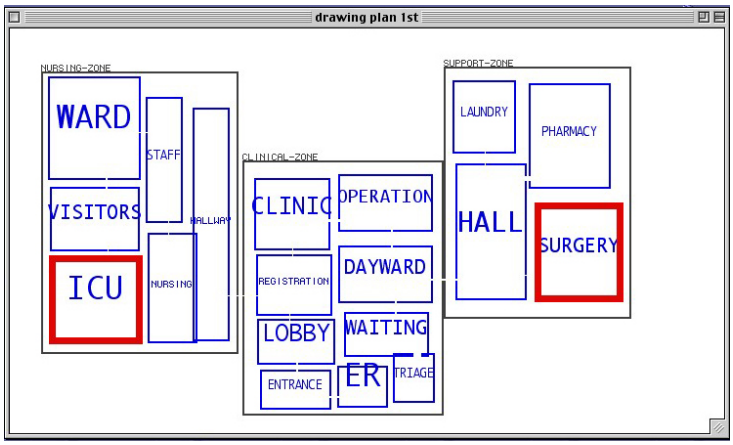
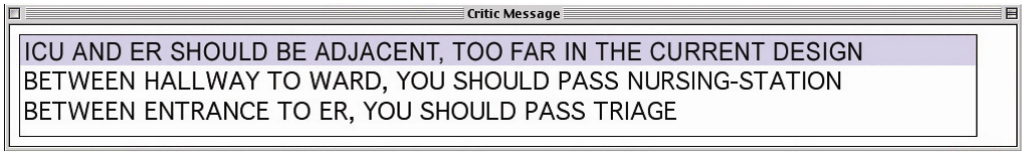


Figure 3.11 Annotated Drawings of Zone Checker (Rectified Mode); Zone Checker highlights the wrongly placed rooms in red color. The boundaries of the ICU and the SURGERY displays in red color.

The Zone Checker shows the designer the locations of the wrongly placed rooms with highlighted thicker lines and also gives a text suggestion to move the rooms to another zone that has the

proper character (Figure 3.11). The verbal and visual critiques are connected: if the user clicks on the second message in Figure 3.12-(a), the Path Checker will display the path from Ward to Hallway in thick red line, as shown in Figure 3.12-(b).

(a)



(b)

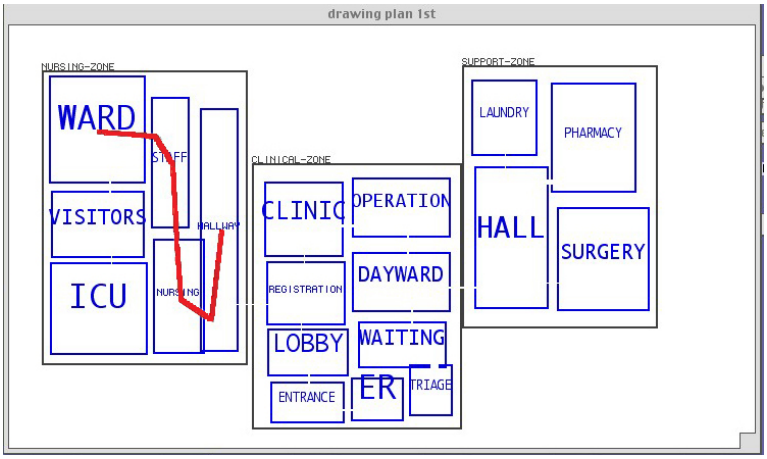


Figure 3.12 Annotated Drawings of Path Checker (Rectified Mode); (a) When the designer clicks the second message of "BETWEEN HALLWAY TO WARD, YOU SHOULD PASS NURSING-STATION", (b) the Path Checker displays the path from HALLWAY to WARD.

3.6.3 Texture-mapped 3D Visualization

The third method for providing design support is a 3D visualization of the space with VRML (Virtual Reality Modeling Language). The stored coordinates and heights of sketch objects are used in generating the 3D models. The *Display Manager* creates the 3D objectprimitives as walls and doors with this information. In addition, each wall of the generated 3D models is texture-mapped with image pictures. If the drawn room is Operation Room, the system finds the pictures of operating room that have same file name to the room name, Operating Room. The pictures are then used to map on the each extruded wall.

Chapter 4

Implementation of Design Evaluator

4.1 DESIGN DATA STRUCTURE

4.1.1 Main Lists of the Design Representation

The design representation in the Design Evaluator system comprises three main lists; *Zone list, Room list*, and *Door list*. These are global variables. Whenever the designer sketches bubbles and lines, the system creates an instance and adds it to one of these lists. The system has another kind of list, *Sketch-glyphs-list.* It stores all sketched points of every glyphs.

4.1.2 Interrelated Sketched Objects

The sketched objects are interrelated. Each object stores a list of the related objects. For example, zone object stores a list of all the rooms that are placed in the zone. The room object stores a list of its doors that connect with other room objects and it also stores the zone object in which it is placed. Conversely, each door object stores the two rooms it connects.

For example, when the designer draws the ICU in the Clinical-zone, the ICU room object points to the Clinical-zone object and the Clinical-zone object adds the ICU to its **`rooms`**. When the designer draws a door between the ICU and the WARD, the door object stores the two room objects, the ICU and the WARD in its variables, **`room1`** and **`room2`**. The ICU and the WARD also add the door object to their individual **`door-lists`**.

Each class is defined as follows. Each object stores a unique identifier. The two variables pt1 and pt2 store the top left and bottom right corners of each object. Each object also stores an id and a

label that is used to display the object in text messages to the designer and to label the floor plans. These variable, pt1, pt2, id, label, sketch-glyphs and are common in the all objects.

(1) Zone Object Class

```
(defclass zone ()
  ((z-id :accessor Z-ID :initarg :z-id :initform (next-z-id))
   (color :accessor color :initarg :color :initform nil)
   (pt1 :accessor pt1 :initarg :pt1 :initform nil)
   (pt2 :accessor pt2 :initarg :pt2 :initform nil)
   (label :accessor label :initarg :label :initform nil)
   (rooms :accessor rooms :initarg :rooms :initform nil)))
```

Zone object class is defined by the above variables. In the **z-id** (zone id), a unique number is generated automatically. The **rooms** variable stores a list of the rooms that are placed in this zone.

(2) Room Object Class

```
(defclass room ()
  ((r-id :accessor roomid :initarg :r-id :initform (next-id))
   (color :accessor color :initarg :color :initform nil)
   (pt1 :accessor pt1 :initarg :pt1 :initform nil)
   (pt2 :accessor pt2 :initarg :pt2 :initform nil)
   (label :accessor label :initarg :label :initform nil)
   (height :accessor height :initarg :height :initform 15)
   (zone :accessor zone :initarg :zone :initform nil)
   (door-list :accessor door-list :initarg :door-list :initform nil)))
```

Room object class is defined by the above variables. In the **r-id** (room id), the successive number is inputted automatically. The designer can change the **height** of each room and this height value is used in generating 3D VRML models. A room object stores the **zone** where the

room object is located in. The room object stores a list of doors (**door-list**) that connect with other room objects.

(3) Door Object Class

```
(defclass door ()
  ((d-id :accessor d-id :initarg :d-id :initform (next-d-id))
   (color :accessor color :initarg :color :initform nil)
   (pt1 :accessor pt1 :initarg :pt1 :initform nil)
   (pt2 :accessor pt2 :initarg :pt2 :initform nil)
   (height :accessor height :initarg :height :initform 0)
   (room1 :accessor room1 :initarg :room1 :initform nil)
   (room2 :accessor room2 :initarg :room2 :initform nil)))
```

Door object class is defined as the variables listed above. The designer inputs and changes the **height** value, which is used in generating 3D VRML models. Each door stores two rooms **room1** and **room2** that are connected by this door.

4.1.3 Path List and the Path Finder Program

After the designer has sketched a floor plan, the system finds all available paths through the building. Each path is a list of rooms. The paths are stored in a global variable, *Path list*. Using these relations of sketched objects, the Checkers obtain the circulation paths by analyzing the sketch floor plan. A path-finder program walks from one room to the next room through the door that connects them. From the second room the path-finder looks for any doors that it has not gone through and continues to explore. This exploring process applies to all rooms and doors recursively using the co-routine functions: ***explore-doors*** and ***explore-rooms***.

Explore-rooms function starts with a room, creates a path, and constructs the visited rooms list. It

calls ***explore-doors*** on each door of the room. ***Explore-doors*** finds a door to enter the next room and calls ***explore-rooms*** to keep exploring the rooms. It removes the previously passed doors (***remove-visited-door-list***) to prevent from leading to the rooms of the visited rooms list. The path-finder program applies ***Explore-rooms*** to all room in room list.

4.2 RULE LIST

The system has a list of rules as global variables. There are four kinds of rules. All rules are stored as a list in the Rule database. The following table 4.1 summarizes all types of rules (see section 3.4.2)with example expressions and explanations.

Table 4.1 All Types of Rules

Checker	Type of Rules	Expression
Path Checker	Room Sequence Requirement	**(<Sequence> <room1> <room2> [<room3>])** e.g. (MUST-PASS-THROUGH ENTRANCE TRIAGE ER)
	Adjacency Requirement	**(<Adjacency> <room1> <room2>)** e.g. (SHOULD-BE-ADJACENT ER INPATIEN-SURGERY)
Zone Checker	Room Placement Requirement	**(<Placement> <Zone> (<Room> <Room>…))** e.g. (MUST-BE-IN NURSING-ZONE (NURSING-STAITON WARD STAFFS-ROOM VISITORS-ROOM …))
Area Checker	Minimum Area Requirement	**(<Minimum-area> <room1> <minimum-size>)** e.g. (MINIMUM-AREA WARD 10000)

4.3 CRITIQUING

The Design Evaluator compares the architectural diagram with the four kinds of rules. These checking operations are as follows. The Design Evaluator has Zone list, Room list, Path list and Rules list. Each rule is applied all relevant paths and the zone. Firstly, the ***find-relevant-paths*** function and ***find-relevant-zone*** function find the relevant paths and zone with each rule. Secondly, ***make-function-call-from-rule*** function generates the lisp function calls. These lisp function calls are evaluated with the arguments: (1) the rooms from a rule and (2) the relevant path or zone. Thirdly, if a certain path or zone violates the rule, the Checkers generate text messages.

4.3.1 Find Relevant Paths and Zones

The system has interrelated objects of path, zone and room lists as well as the built-in rules. The current design is compared with the built-in rules, when the critiquing command is issued. The Design Evaluator system finds relevant paths and zones. Here is the definition of the ***find-relevant-path*** function**.**

```
(defun find-relevant-paths (second rule) (third rule) path-list
  (mapcar #lambda (path)
     (if (and (find (second rule) path)
              (find (third rule) path)) path))
  path-list)
```

The ***find-relevant-paths*** function takes the room names in the path rule and finds the paths which have these rooms in the path. For example, function takes the rule of (SHOULD-BE-ADJACENT

ER INPATIENT-SURGERY) and checks with current configuration path list to return a subset of the path list that contains all the rooms listed in the rule, (ER INPATIENT-SURGERY). As section 3.4.2, the above rule is a list of three items. The first item is the expression for the rule (SHOULD-BE-ADJACENT). The second item of the rule list expressed as **(second rule)** is. The third item **(third rule)** is. The ***find-relevant-paths*** function takes the argument of the 2room name (ER, INPATIENT-SURGERY) like the following function and checks against the path list and return all paths that have ER and INPATIENT-SURGERY.

```
(find-relevant-paths ER INPATIENT-SURGERY path-list
  (mapcar #lambda (path)
     (if (and (find ER path)
              (find INPATIENT-SURGERY path)) path))
  path-list)
```

The ***find-relevant-zone*** function takes the name of zone in the zone rule, finds the zone object with that name in the zone list.

```
(defun find-relevant-zone (second rule) zone-list
  (find (second rule) zone-list :test 'string-equal :key 'label))
```

For example, ***find-relevant-zone*** function takes the zone rule of (MUST-BE-IN NURSING-ZONE (NURSING-STAITON WARD STAFFS-ROOM VISITORS-ROOM …)) and find the Zone object whose label is NURSING-ZONE. The second item of this rule, **(second rule)** is NURSING-ZONE. Therefore, this function will return the zone object which has the "NURSING-ZONE" label.

4.3.2 Apply Rules to the Relevant Paths and Zones

The Design Evaluator has three checkers: Path Checker, Zone Checker and Area Checker. Path Checker evaluates two kinds of requirements: room sequence requirement and adjacency requirement. The ***make-function-call-from-rule*** function takes a path rule and creates a lisp function call. For example, one rule is (SHOULD-BE-ADJACENT ER ICU). The ***should-be-adjacent*** function takes two rooms and a list of paths. It checks whether two rooms is adjacent in the paths. The following code shows how the ***make-function-call-from-rule*** function works.

```
(make-function-call-from-rule (rule relevant-path)
  (list (first rule)(second rule)(third rule) relevant-path))
```

For example, the ***make-function-call-from-rule*** function can take the rule of and the relevant-paths found by the function and returns a list of function calls. In this function, **`(first rule)`** is SHOULD-BE-ADJACENT, **`(second rule)`** is ER and **`(third rule)`** is ICU. The function ***find-relevant-paths*** returns a relevant path (e.g. ····· ENTRANCE – ER – HALLWAY – ICU ·····). Therefore, it creates the function-call **(should-be-adjacent ER ICU relevant-paths)**, which is then evaluate in the Path Checker.

Path Checker calls ***make-function-call-from-rule*** function and evaluates the called ***(should-be-adjacent ER ICU (ICU - VISITORS-ROOM - WARD- STAFF-STATION - NURSING-STATION - HALLWAY - LOBBY - ENTRANCE - ER))*** function. These codes enable the design evaluation to make many function calls effectively, regardless of the number and kinds of rules. For each relevant paths found, it will produce a function call. For example, if the rule (SHOULD-BE-ADJACENT ER ICU) is applied to path and it checks out then it goes on to apply

the same rule to the next path. When there is a violation, the function will produce a design critiques.

The Zone Checker works in the same way. The ***must-be-in*** function takes zone rule and the relevant zone and compares the room list of rules with the rooms which the relevant zone has. For example, the Zone Rule for CLINICAL-ZONE that describes the room placement requirement is **(must-be-in CLIINICAL-ZONE ("REGISTRATION-OFFICE" "OPERATION-ROOM" "ICU" "INPATIENT-SURGERY" "PHYSICAL-THERAPY-ROOM" "REGISTRATION-OFFICE" "PACU" "PREOPERATIVE-AREA" "TRIAGE" "ER" "TREATMENT" "DAYWARD" "CLINIC-FOR-OUTPATIENT")).** When Zone Checker calls ***make-function-call-from-rule*** function, it evaluates the function of ***(must-be-in Clinical-zone ("REGISTRATION-OFFICE" "OPERATION-ROOM" ….)).*** The function ***must-be-in*** takes the room objects that are located in the ***relevant-zone*** (Clinical-zone) and the room list of the rule. If there is the difference between two lists, function ***must-be-in*** generates the critique.

The Area Checker checks the minimum size of the sketched rooms. It takes the area rule, **(min-area ER 5000)** and finds the ER object in the floor plan, if it exists. The MIN-AREA function then calculates the area of the bubble that represents the ER and compares it with the minimum size predefined by the area checking rule and returns a true(T) or false (Nil) statement. If the current ER is smaller than the minimum requirement (5000) in this case, it then returns a function to generate critiques . The ***make-function-call-from-rule*** function calls and evaluates the MIN-AREA function.

4.4 DISPLAY CRITIQUES

4.4.1 Verbal Critiques

The Checkers generate verbal critiques and display all of the critiquing text in the critique window. Checkers include the ***should-be-adjacent*** function, the ***must-pass-through*** function, the ***must-be-in*** function and the ***minimum-area*** function.

Firstly, the ***should-be-adjacent*** function takes the argument of **`room1`**, **`room2`** and **`relevant-paths`** and then generate a verbal critique.

```
(defun should-be-adjacent room1 room2 relevant-paths
  (if (room1 and room2 is not adjacent)
  then (create-critique "room1 AND room2 SHOULD BE ADJACENT, TOO FAR IN
THE CURRENT DESIGN")))
```

In other words, if the ICU and the ER are placed apart in the design, the ***should-be-adjacent*** function will generate the text critique of "ICU AND ER SHOULD BE ADJACENT, TOO FAR IN THE CURRENT DESIGN".

Secondly, the ***must-pass-through*** function creates text critiques by taking the arguments of **`room1, room2,`** and **`room3`** as defined by the rules and the relevant-paths to generate critique of room sequence requirement.

```
(defun must-pass-through room1 room2 room3 relevant-paths
  if (path does not keep the sequence of room1-room2-room3)
   then (create-critique "BETWEEN room1 TO room3, YOU SHOULD PASS
room2"))
```

Thirdly, the ***must-be-in*** function creates text critiques by checking each room in the zone. For example, comparing the **(must-be-in NURSING-ZONE rooms)** with the current list of rooms in

this zone, (**rooms NURSING-ZONE)** will return any different room not described in the rule. If there is any different room, the function generates the critique with the desired zone which can be found in the rule, **NURSING-ZONE.**

```
(defun must-be-in rooms relevant-zone
  (if (set-difference rooms (rooms relevant-zone))
     then (create-critique "Wrongly-placed-room SHOULD BE PLACED IN the
    desire zone")))
```

Finally, the ***minimum-area*** function creates text critiques when the requirements were not met. In the following function, **min-area** is defined by the rule, for example, (MINIMUM-AREA SURGERY 6000). The **relevant-room** is the room object with the label, "SURGERY". The function then calculates the area size using pt1 and pt2 of that room object and compares the size with the **min-area.** If the calculated area is smaller than the **min-area**, the function generates the text critique that the room is too small and states the minimum area requirement.

```
(defun minimum-area min-area relevant-room
  if ( > min-area (area size of relevant-room))
     then (create-critique "relevant-room IS TOO SMALL, THE MINIMUM AREA
    IS min-area"))
```

4.4.2 Visual Critiques

Checker functions also store the problematic paths and the wrongly placed rooms. The system operates with two functions to provide the designer with visual critiques. These two functions include the ***display-path*** and ***display-wrong-rooms***. First, the ***display-path*** function takes the first room in the problematic path and computes its center point. It then draws a line from the center of

the first room to the center point of door it shares with the second room. The function then draws a line from center of the door to the center of the second room. The ***display-path*** function continues to apply this work to all rooms in the path. At the same time, the ***display-wrong-rooms*** function takes the wrongly placed rooms and highlights the rooms in red on the drawing.

4.4.3 3D Visualization

The Design Evaluator system stores the boundaries of all the sketched rooms and doors. The system generates VRML codes with the stored coordinate positions and heights. These VRML codes generate the geometries. For example, room object has pt1 (top-left corner) and pt2 (bottom-right corner). The system calculates all four points of a room rectangle and generates the four walls with the height of that room object.

Then the system uses real photos from clinical rooms to use texture-mapping images for the walls the room. These photos have same names as with rooms. For example, if the designer draws a room labeled as OR, the Design Evaluator system finds the jpeg file that has same name, OR to map onto the four walls of the 3D room model.

4.5 REPAIR THE DRAWING (DESIGN EVALUATOR'S EDITING OPERATION)

When the designer gets feedback from the Design Evaluator, the designer might change the current design alternative. S/he can move and erase the rooms to fix problems in the current design. The Design Evaluator has two modify functions; move and erase. When modifying the room, the relations among objects may be changed, because the sketched objects are interrelated with each other. For example, two rooms that were adjacent, or connected may no longer be adjacent or connected. Therefore, the system recomputes the relations of objects and update all

the database.

4.5.1 Erase Operation

The erase function captures the point location where the designer clicks on the drawing window. If the point is in a certain room object, the system erases this room. It then recomputes the related objects with this room. For example, if the system remove the ENTRANCE object specifically, it also must remove ENTRANCE from all the doors that currently refer to it. If the designer wants to remove the ENTRANCE, the erase function must perform a series of removing activities as shown in Figure 4.1:

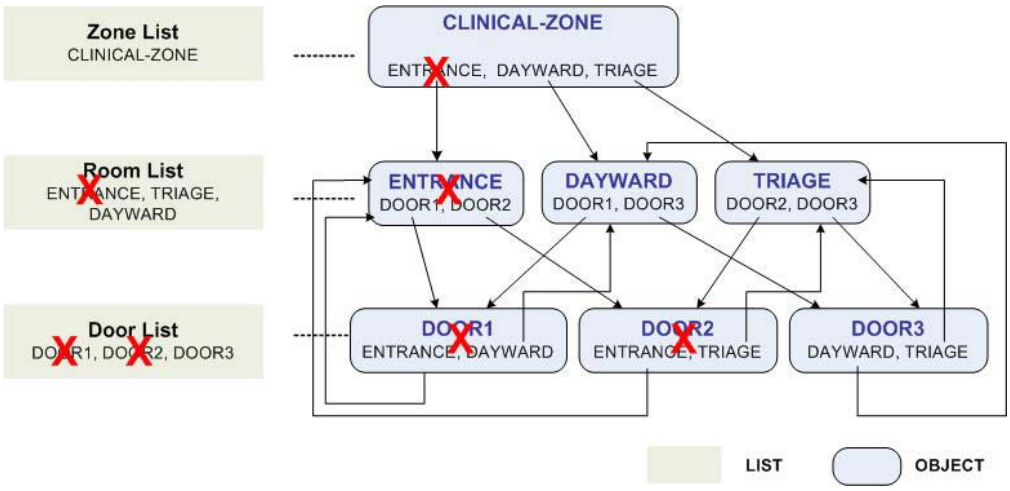


Figure 4.1 A series of activities for removing the ENTRANCE room

(1) Removing this room from the room list and sketched glyphs list

The function finds the room object that contains the designer's input point. If the point is in the ENTRANCE room, the function removes the found room from the whole room list. It finds the sketch glyphs list of ENTRANCE room, then removes it.

(2) Erasing this room slot from its zone

Zone object (CLINICAL-ZONE) has the room objects (ENTRANCE, TRIAGE, and DAYWARD). The erase function should remove the ENTRANCE room from the zone that contains this room.

(3) Erasing door objects

Room object also is related with Door object, because door object has room objects as slots. Therefore, the function finds the related doors, ***(door-list room)*** and removes the DOOR1 and DOOR2 from Door list .

4.5.2 Move Operation

If the designer clicks a certain room to move it to another position, the move function chooses the room and moves it to the take new position specified by a second click. Figure 4.2 describes a series of activities of the move function.

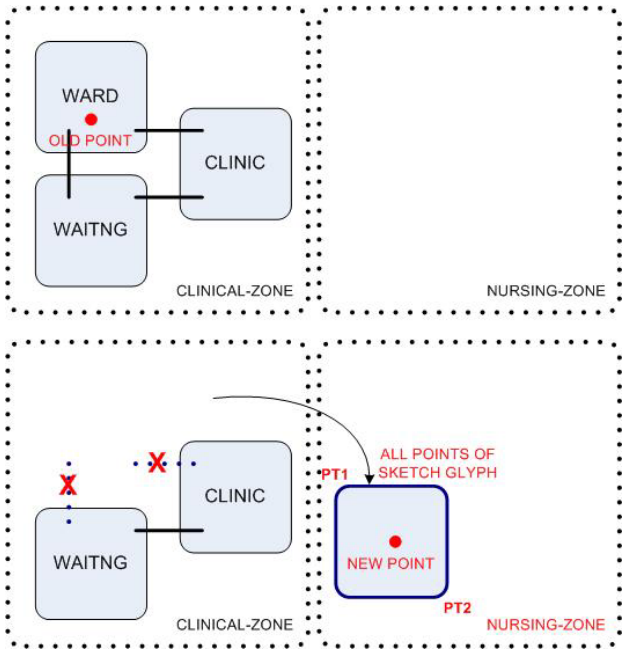


Figure 4.2 A series of activities for moving WARD room

(1) Calculating the new coordinate position with the new taken point

The moving function calculates the new coordinate position using the movement information from the distance between the derived old point and the new point. It calculates coordinates and generates new coordinates: (1) PT1 and PT2 of a room object and (2) all points of the sketched glyphs.

(2) Re-computing which zone the room is place in

Now the room is in a new zone. Therefore, the zone information should be updated. The WARD was in orginally the CLINICAL-ZONE, but the room is currently in the NURSING-ZONE. The moving function re-computes where the room is and puts the room into the slot of the NURSING-ZONE object and delete it from the old zone, the CLINICAL-ZONE object.

(3) Removing the door objects that are related to the moved room

The moving function erases doors that are connected with the moved room. For example, two doors of WARD would be erased from the slots of WARD and from the door list.

4.6 SAVE-LOAD OPERATION

The Design Evaluator can save and later reload the design. The ***Save*** function writes all objects to streams, or files. This function calls the ***write-zone-to-stream*** function, ***write-room-to-stream*** function, and ***write-door-to-stream*** function. For example, the ***write-room-to-stream*** function writes an individual room to a file. However, when the system later loads these files, the interrelations of the sketched objects are lost and must be recomputed. This is because each object has the related other objects as a slot. Therefore, when the ***Save*** function works, ***write-zone-to-stream*** function, ***write-room-to-stream*** function, and ***write-door-to-stream*** function write the inherent object-id instead of the object.

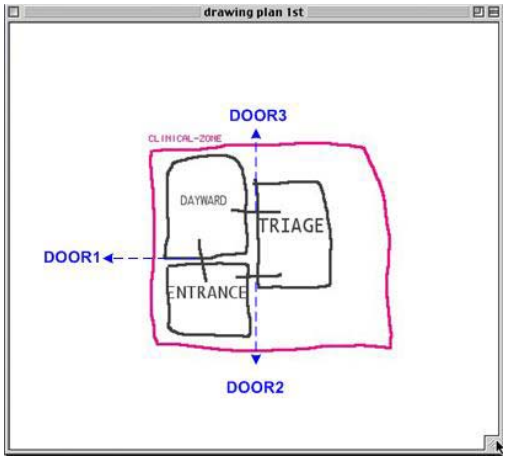


Figure 4.3 In this simple diagram, all elements are interrelated with each other. For saving operation, the save function saves object-id instead of object.

For example, in Figure 4.3 TRIAGE Room object has zone object and door objects as

follows. **#<ZONE #xEE6999E>** indicates Clinical-zone, and **(#<DOOR #xEE7EEBE> #<DOOR #xEE7EAD6>)** indicates two doors that connect to DAYWARD and ENTRANCE.

```
#<ROOM #xEE7B746>
Class: #<STANDARD-CLASS ROOM>
Instant Slots
Room-ID: 1
Label: TRIAGE
PT1:10813698
PT2:18284882
…
Zone: #<ZONE #xEE6999E>
Door: (#<DOOR #xEE7EEBE> #<DOOR #xEE7EAD6>)
```

However, when the system loads the saved streams, the system cannot interpret the object such as **#<ZONE #xEE6999E>**. Therefore, the inherent object-ID and the ***Save*** function are written as follows.

```
#<ROOM #xEE7B746>
Class: #<STANDARD-CLASS ROOM>
Instant Slots
Room-ID: 1
Label: TRIAGE
PT1:10813698
PT2:18284882
…
Zone: 1
Door: (3 2)
```

After reading the streams, the system makes the connections of the objects. In other words, the system finds the object with the object ID and replaces the object ID with the found object.

In this chapter, we mentioned how the Design Evaluator system is implemented from the fundamental data structure to several functions. After explaining the main list and all object class

in the data structure, we describe two things; 1) interrelations of the sketched objects and the explored paths and 2) the summary of the built-in rule list. We then describe the critiquing process of Checkers with the several functions and the display of three kinds of critiques. Finally we explain the editing operation such as erase and move function and save-load functions.

Chapter 5

Related Work

Design Evaluator incorporates two issues of the 2D sketching as an interaction medium and the floor plan evaluation with critiquing. Therefore, this chapter of the thesis describes the three kinds of related work: roles of design drawings: sketch-based design system and critiquing system. In the first section, we describe three roles of design drawing. We then examine where the Design Evaluator is located in the existed computation support systems. We describe two kinds design support systems: sketch-based design system and critiquing system.

5.1 ROLES OF DESIGN DRAWINGS

Designers draw sketches and diagrams to represent design ideas in the early stage of designing. Design studies researcher notes that drawings are an essential part in designing. In first section, we categorized the roles of drawing into three kinds: externalizing, communicating, and reflecting on design ideas.

Firstly, drawing is a way to externalize designers' ideas. Designers usually draw diagrams and sketches using graphic elements to represent the visual images in their mind. Often times designers also express their design ideas using words. However, drawing conveys visual and spatial ideas directly and better than spoken or written words (Tversky, 2002). Drawings help designers to understand and envisage design ideas, because spatial relations of the graphical elements are directed available in the drawing.

Secondly, drawings facilitate communication among the designers as well as themselves. Designers also communicate with other people, co-workers and stakeholders with drawings. While designers work together in the design process, they need to share information to generate design solution with a common concept (Laseau, 1989). The visual images of marks on the paper are conveyed to designer's mind through the eyes. These images help designers to see the unintended discoveries as well as to check the design ideas during the drawing process. Drawing serves as a form for designer's self-communication.

Finally, researchers in the field of design studies have identified the role of freehand design drawings (i.e. sketches and diagrams) as material that stimulates reflection in the early design stages. Schön describes designing as 'reflection-in-action' - designers go through the action of generating a design solution, evaluating it, reflecting on and changing it. He argues that drawing is essential as a tool in this reflecting process (Schön, 1985). Designers use drawings to externalize design ideas and then to reflect and to develop their designs further. Through examining and interacting with the drawings, designers develop and modify their design ideas. Designers must see the visual image on the drawing (Goldschmidt, 1991) to make a decision, to add a new design idea, or to modify the design (Laseau, 1989). Schön argues that designers perform 'seeing-moving-seeing cycles' in designing. In this cycle, 'seeing' is the interpretation of a drawing that is composed of graphical symbols; it induces the designers to have a conversation with themselves about the design ideas that they have recorded in the drawing (Schön and Wiggins, 1992). This feedback initiates an action, resulting in adding, moving, or removing design symbols in the drawing.

5.2 SKETCH-BASED DESIGN SYSTEM

Below we briefly review related work of computational sketch system in two categories: 1) sketch recognition and 2) knowledge capture and 3D visualization.

5.2.1 Sketch Recognition and Knowledge Capture

The Design Evaluator is part of our larger research agenda on intelligent support for design sketching (Gross and Do 2000). The Electronic Cocktail Napkin system (Gross 1994) and its various extensions explored the applications of sketch recognition in various knowledge-based design tasks (Figure 5.1). For example, if the user draws a stack of boxes or spiral, the system can recognize the diagram as a plan or elevation of Wright's Guggenheim museum.

Figure 5.1 Electronic Cocktail Napkin support sketch objects capturing of and simulated drawing environment

Various sketch-based systems have been developed to support design. These are appropriate for early design stages, because sketches make it easier to quickly explore design solutions. For creative design work, computationally enhanced sketching might offer additional features, based on reasoning about the sketches. Here we consider the knowledge capture of sketch-based systems.

SketchIT (Stahovich, 1996) is a system for conceptual design of mechanical devices such as hook and pushrod. SketchIT identifies the parts and simulates the system's behavior. SketchIT analyzes the behavior of parts and then produces the desired behavior.

sKEA (Sketching Knowledge Entry Associate) (Forbus and Usher 2002) is designed for capturing knowledge from sketches. sKEA can acquire several kinds of information from the sketches; what the glyphs mean (semantic information), where they are placed (positions), what relations one glyph has with others and which glyphs are similar conceptually and visually.

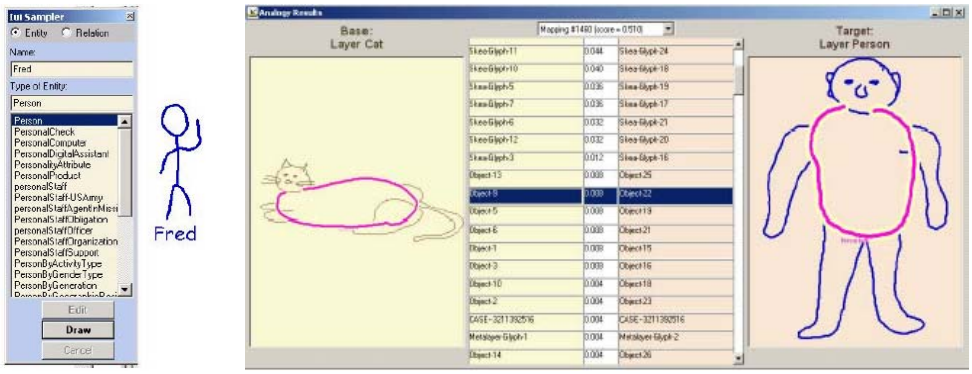


Figure 5.2 sKEA (Sketching Knowledge Entry Associate); (a) Interpretation of glyphs: recognition of visual symbols, linguistic labeling, and composition of meaning from interpretations of more primitive parts (b) Visual and conceptual analogies: the rounded body of a cat and the rounded human torso

For example, sKEA matches the rounded body of a cat and the rounded human torso (Figure 5.2). This matching capability can suggest what glyphs would be added and where they would be added, because people usually share graphic conventions and tend to sketch in the same way.

5.2.2 3DVisualization

Other sketch-based systems support the designer with representing designs with 3D visualization. For example, VR Sketchpad (Do 2001) (Figure 5.3) extended this work in the direction of creating 3-D models from 2-D sketches, which underlies the 3D VRML model display component of our

current work in Design Evaluator. The Sketch VR recognizes the simple geometric shapes like lines and circles and the several symbols. If the designer draws several lines and circles, the system recognizes these shapes as walls and columns and then creates instant 3D views. In addition, the system supports furniture placement. For example, the designer draws TV or Table, Sketch VR places the 3D models of TV or Table on the 3D space.

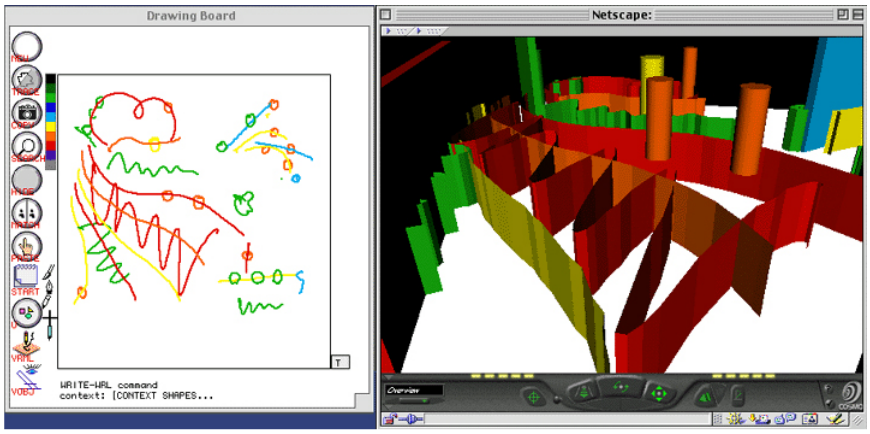


Figure 5.3 VR Sketchpad: VR Sketchpad transforms 2D drawing into 3D VRML

Teddy (Igarashi 2000) enables a designer to quickly generate a three-dimensional model from a sketch. Teddy generates three-dimensional spherical objects with a polygonal mesh presentation which is useful, for example, for early design stage of character animation (Figure 5.4).

Figure 5.4 Teddy: (a) Teddy in Use, (b) The painted models created using Teddy and painted using a commercial texture-map editor

5.3 CRITIQUING SYSTEM

A critiquing system is an effective way to use computer knowledge bases, because it provides feedback for designers to improve their design (Silverman 1992), and reduces designer's cognitive load. Critiquing systems typically have a series of rules or procedures for evaluating a design solution and identifying problems (Fisher et al. 1991).

Several critiquing systems have been investigated in various design areas; kitchen design (KID, CRACK and PREDIKT), hardware and software design (Critter, VDDE and Petri-NED) and architectural design (code and floor plan checking).

KID (Knowing-in-Design) (Nakakoji 1993) and CRACK (A Critiquing Approach to Cooperative Kitchen Design) (Fisher and Morch 1988) support designing a simple kitchen floor plan with text messages. These systems provide critiquing messages for problematic aspects such as a poorly placed appliance or an incorrectly sized work triangle and offer the designer examples of kitchen layout that might be appropriate (Figure 5.5).

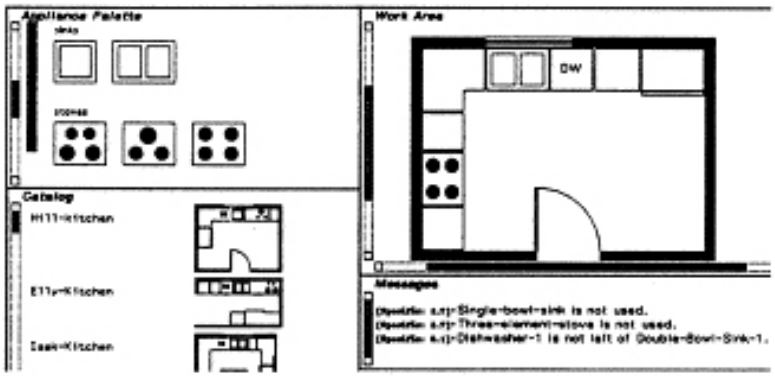


Figure 5.5 KID (Knowing in Design) and CRACK (A Critiquing Approach to Cooperative Kitchen Design)

Similarly PREDIKT (Oxman 1992) is an expert system intended for generating and evaluating kitchen design. It interprets designs with positional and typological knowledge. The user can

select the reflection modes: critique generation or design generation.

In hardware and software design, critiquing systems are also used. Critter (Kelly 1984) aids digital circuit design with critiques about operating speed, timing robustness, operating speed and circuit sensitivity. VDDE (Voice Dialog Design Environment) (Repenning and Sumner 1992) supports the design of phone-based user interfaces. If it detects conflicts between the voice dialog design and the VDDE embedded rules, then it displays critiquing messages in the message pane in a prioritized order. Most systems provide text critiquing message in a separate text window. However Petri-NED (Stolze 1994), a design environment to support the design of Petri nets, provides visual critiques overlaid on the Petri-Net design diagram instead of text critiques (Figure 5.6).

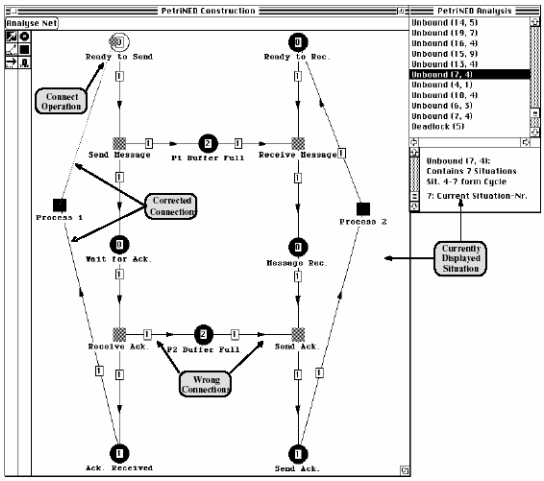


Figure 5.6 Petri-NED shows visual critiques of the design of Petri-Nets

Code checking and floor plan checking are long-standing problems for developing software intelligent CAD. Checkers are typically not used in design generation process; rather these checkers are applied to the final design solution for fixing the problems. Examples include code

checking (Woodbury et al. 2000), a floor plan checker (Kalay and Sequin 1995) and the commercial Solibri Model Checker (Solibri inc. 2003). Checking systems typically is used to the final design not in design process. However, these checking systems are similar to a critiquing systems in that these systems interpret a design and identify problematic parts of the design. Most code checking or floor plan checking systems need drawing editors for checking. SMC checks an architectural 3D model with constraint sets such as model integrity, material life-cycle, building code, and physical security. Although SMC provides useful checking features, a user would have trouble getting critiques in early design because SMC requires extremely detailed modeling work before the checking process can begin (Figure 5.7).

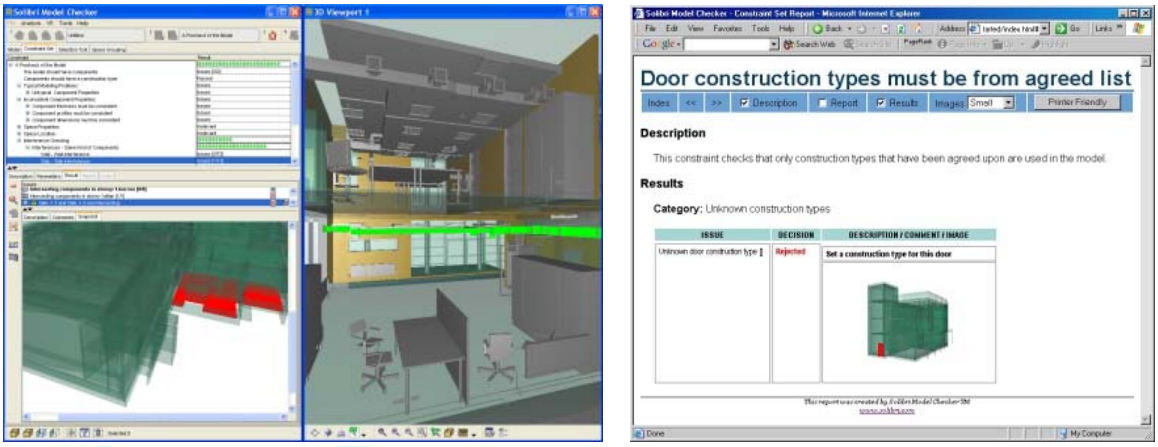


Figure 5.7 Solibri Model Checker: (a) Overview of the User Interface, (b) Decision, Comment and Snapshot

5.4 SUMMARY AND DISCUSSION

We examined three rules of designer's freehand drawings: externalization, communication and reflection. According to the design studies about the freehand sketching, it is a reflection or interaction medium. For stimulating this designer's reflection-in-action cycle, we provide the designer with the design feedback as critiques. Therefore, this thesis is located between sketch-

based design system and critiquing system. In the previous section, we describe the design knowledge capture and 3D visualization of the sketch-based design systems and performances and display methods of several critiquing systems.

The Design Evaluator system captures the spatial relations and functional relations from the freehand diagram. It checks the captured information with the rules, then displays critiques in three ways.

Chapter 6

Discussion and Future Work

6.1 SUMMARY

We have proposed that early design critiquing, integrated into a freehand sketch design system would be a useful tool that is effective to identify errors, we built a working prototype, Design Evaluator, to demonstrate the potential of this approach to knowledge-based design. Unlike other critiquing systems, which employ structured drawing editors, Design Evaluator incorporates critiquing in a sketch-based design system. We gain two advantages by doing so. One advantage is the improvement of the sketch design system by providing design critique directly on the work place where drawings are produced. The designers often use sketches to interact with mental imagery, analyze and examine design ideas. This mental imagery from the sketches triggers generation of new design solutions. The offered graphical critiques by the Design Evaluator system could potentially stimulate mental imagery from sketches. Second, when the designer becomes aware of errors in the design from critiques, he/she would likely to want revise and improve it. Sketching makes design revision easy and allows the designer to explore other design ideas quickly.

Design Evaluator is embedded with built-in rules about spatial arrangement. We have proposed what design constraints can be expressed as rules. The current Design Evaluator system has four kinds of rules; room placement rules, proper room sequence rules, adjacency rules, and minimum area rules. When conflicts happen between the sketched floor plan and the built-in rules, Design Evaluator brings the designer to the attentions these conflicts. This gives the designer a chance to

improve and to revise the design and eliminates problems that might otherwise remain undetected or costly to fix in the later stage of design.

Design Evaluator delivers design critiques in three ways: text critiques, annotated drawings, and 3-D annotated walk-throughs. Design Evaluator is intended for architects, who communicate their designs with themselves and other people using text and graphical modes. Communicating design information with only one mode can be problematic and the combination of displaying graphical critiques with text critiques is more effective. For example, if the system gives the only text critiques, it poses significant cognitive load to the designer. For example, if the Design Evaluator only prints out a complaint about a path problem with highlighting or displaying the problem, the designer will have to go through all possible paths on her design to search for the problematic circulation path. This is time consuming and error-prone. Therefore, visual critiques are provided in the Design Evaluator the right there on the design drawing, so that the cognitive load is significantly reduced.

Design Evaluator supports sketching design with design critiques. When the designer receives the critiques, s/he will then mostly evaluate the design and revise it. A sketch-based system makes the revision of design easy and it supports a designer to explore ideas continuously. The Design Evaluator environment supports designer's reflection-in-action process effectively, because the offered critiques stimulate the design cycle.

In the following sections, we describe several future research directions in four issues: Rules, Sketching system, displaying method of critiques and possible Design Evaluator system for other design domains.

6.2 RULES AS DESIGN KNOWLEDGE

6.2.1 What Design Knowledge can be translated into Rules?

The Design Evaluator uses rules expressed in text as the design knowledge for critiquing the proposed design. These text rules are translated from design constraints, design conventions or rules of thumb. We have shown how these design constraints can be expressed as the text rules. However, the current research has not investigated what design knowledge can and cannot be translated into text rules. If some knowledge cannot be expressed in rules, then we would like to understand this. Or, if some design knowledge can not be translated into rules, we would like to investigate how to represent this design knowledge into an intelligent design system. Some design knowledge may be more easily expressed graphically, then we can input the rules in the drawing as well, instead of just the text.

6.2.2 Other Possible Rules for Floor Plan Checking

It is quite easy to add new kinds of Checkers into Design Evaluator and we would like to extend Design Evaluator's knowledge base by adding different kinds of checkers. These could include checking for circulation conflicts, the capability of calculating dimensions of sketched objects for ADA (Americans with Disabilities Act) Compliance, and support for floor plans of different stories (basement, 1F, 2F, elevator, stairs, exit, etc.) to allow checking of vertical circulation.

One potentially useful rule is to check for circulation conflicts. Architects need to achieve consider circulation conflicts in architectural planning, especially in a complex building design like a hospital. The designer should identify and separate the movements of different user types and resolve any conflicts among them. A hospital has three main kinds of building users: inpatients, visitors, and medical staff. Mixing the movement flows of these three types could cause building function inefficiency. In addition, unexpected accidents may result from it. For

example, a hospital visitor to the inpatient ward should gain access from the entrance to the ward without having to pass through the operation room.

Another useful addition will be to support checking of the vertical circulation of several floors. The current Design Evaluator checks only one floor. However, if the Design Evaluator could consider vertical circulation and vertical zoning, this would be more powerful and general in scope. Let's consider a physical therapy room for disabled persons as an example. A physically challenged patient needs to access the room using a ramp or an elevator. A hospital design will improper placement of rooms, ramps or elevators may cause problems in the real use. Extending the Design Evaluator to consider vertical circulation would enable it to check for this kind of circulation problems or errors. In addition, vertical zoning would be helpful in the design process, because the designer often zones a building vertically (in section), not horizontally (in plan).

The third type of rule checking we like to add is dimension checking for disabled access. The current Design Evaluator's dimension checking function checks only the areas of the rooms. If the system would recognize the widths of the circulation elements such as stairs, and hallways, this could help the architect to check on the accessibility of hallway, door, elevator and rooms for ADA compliance.

6.3 EXTENSIONS TO THE SKETCHING INTERFACE

6.3.1 Multiple Floor Plans

We already mentioned earlier the advantages for the Design Evaluator to recognize the multiple levels floor plans for vertical circulation checking rules (Section 6.2.2). Currently our system only supports a single level floor plan design. The extension to provide support for multiple floor plans in Design Evaluator could also be used to compare the performance of different design options of the same floor against the same design criteria rules.

6.3.2 Sectional Drawing

On other extension for the Design Evaluator is to support sectional drawing design. This could be useful for checking on ceiling height clearance, lighting distribution, and visual access. It would not be hard to add this extension alone. It would be interesting to link the plan diagram with the sections as well.

6.3.3 High-Level Recognition

We plan to strengthen the Design Evaluator's sketch recognition to include high-level object relationships such as "contain", "lines-connect" and "small-size". This high-level recognition will be helpful in automatic detecting of zone-room relations. The current system recognizes which zone the room is located in and which two rooms are connected with each other with simple. This low level recognition is achieved through designer's declaration of checking the radio-button. If the system has more powerful recognition and inference capabilities, it could relieve designers from this declaring task.

For example, when one relatively large bubble contains several bubbles, the system could recognize the former as the zone and the latter as the rooms. The lines between two rooms also can be automatically recognized as doors. If the system could do this recognition, the designer would not have to do the declaring process. After a diagram is drawn, the system could infer which object is zone, room or door using the spatial relations like "containing (relation between the zone and the room)", and "connecting (relation between the room and the door)".

Another benefit of adding recognition abilities to Design Evaluator is to recognize different symbols representing building elements. If the system holds the relations of raw glyphs as well being able to recognize whether a shape is a rectangle or a circle, and what spatial relationships among them such as "overlapping", "containing" and "next to" reduce the need to label the

drawing. For example, when the designer draws two rectangles and circle intersecting the boundary of two rectangles, the system could recognize these elements as two rooms and a door. In addition, the system can interpret that one room is located to the right of the other room and that the overlapped door with two rooms connects the two rooms.

Another logical extension to the Design Evaluator system is to include database of domain knowledge for design critiquing. If for example, the designer labels the two rectangles described above as BEDROOM and LIVINGROOM, the system can load the architectural domain knowledge of residential housing for critiquing.

6.4 DISPLAY METHODS OF CRITIQUES

6.4.1 Display the Critiques in 3D Spaces

The current 3D VRML visualization only shows the texture-mapped 3D primitives. However, we can easily provide visual critiques about improper path on room placement on the 3D scene. For example, we can print the room label on the problem path on the floor surface on as billboard object suspended in the space that would be visible from any viewing angle. Besides displaying the path line in the 3D space, the system can also show the problem path on desired sequence as a series of walk-through sequence picture derived from the 3D model or a sequence of view positions embedded in the 3D VRML to guide the view through the space. It will be powerful for the architect to recognize the spatial arrangement in 3D space.

6.4.2 Spoken Critiques

The current text critiques are displayed in the critique window. If the system can speak out the generated verbal feedback to the designer, it can be an effective way to deliver the design critiques. Like the instructor's critiques in architectural studio, the spoken critiques will be

helpful to make the designer reconsider her/his design for avoiding errors. We could also collect and create a databases of famous architect's design critiques in the form of audio, text, video, sketch and gestures.

6.5 OTHER DOMAINS OF SKETCH-BASED CRITIQUING – USER INTERFACE DESIGN

The current Design Evaluator operates in the specific domain of architectural floor plan checking. However, the idea of offering critiques on design sketches can be applied other domains as well. For example, we can easily extend Design Evaluator to support critiquing of user interface design and web page design. The Design Evaluator environment could support the addition or changing of the rules for checking the designs easily, it would be easy to modify the rules in Design Evaluator to be suitable for checking other spatial design criteria for other domains.

In a user interface design critiquing, if for example, two related interface objects are too far apart on the screen, Design Evaluator can interpret the layout of the interface and offer critiques from the predefined rules. For example, the system might detect the problems of too many buttons on the screen or that the configuration of the interface objects being too far apart may increase the time for the task performance.

The Design Evaluator can also check containment requirement such as information organization in different zones or columns on a screen interface or a web page. It could also simulate the path of user's clicking actions of a series of buttons of an interface or the eye movements browsing a web page.

References

Do, E.Y-L.: 2001, VR Sketchpad – Create Instant 3D Worlds by Sketching on a Transparent Window, *CAAD Futures* 2001, Kluwer Academic, pp:161-172

El Croquis: 2002, *Steven Holl*, El Croquis

Goldschmidt, G.:1991, The Dialectics of Sketching, *Journal of Creativity Research* 4(2), pp:123-143

Graves, M.: 1977, The Necessity for Drawing: Tangible Speculation, *Architectural Design* 6 (77), pp:384-394

Gross, M. D.:1986, *Design as Exploring Constraints*, Ph.D. Dissertation, Massachusetts Institute of Technology

Gross, M. D.: 1994, The Cocktail Napkin, the Fat Pencil, and the Slide Library, *Association for Computer Aided Design in Architecture (ACADIA)* 1994, pp:103-113

Gross, M. D. and Do, E.Y-L.: 2000, Drawing on the Back of an Envelope: a framework for interacting with application programs by freehand drawing, in *Computers and Graphics Journal* 24, pp: 835-849.

Fischer, G. and Morch, A.: 1988, A Critiquing Approach to Cooperative Kitchen Design, Proceedings of *the International Conference on Intelligent Tutoring Systems*, Montreal, Canada, 1988, pp: 176-185

Fischer, G. and Lemke, A. and Mastaglio, T.: 1991, The Role of Critiquing in Cooperative Problem Solving, ACM Transactions on Information Systems, Vol. 9, No. 3, 1991, pp: 123-151

Fish, J. and Scrivener, S.: 1990, Amplifying the Mind's Eye: Sketching and Visual Cognition, *Leonardo*, 23(1), pp: 117-126

Forbus, K. and Usher, J.: 2002, Sketching for Knowledge Capture: A Progress report, *International Conference on Intelligent User Interfaces Computer-Aided Design of User Interfaces* '02, San Francisco, CA, USA

Hu, X., Pang, J., Pang, Y., Atwood, M., Sun, W., Regli, W.:2000, A Survey on Design Rationale: Representation, Capture and Retrieval, *American Society of Mechanical Engineers (ASME) Design Engineering Technical Conferences*

Igarashi, T., Matsuoka, S. and Tanaka, H.:1999, Teddy: A sketching Interface for 3D Freeform Design. *SIGGRAPH 99 Conference* Proceedings, pp: 409-416

Ishizaki, S.,:2002, *Improvisational Design*, The MIT Press

Kalay, Y. and Sequin, C., Tools Development and Use in a Multi-Disciplinary Collaborative Computer-Aided Design Studio, *CAAD Futures* '95, Singapore, pp: 21-35

Kelly, V.: 1985, The Critter System: Automated Critiquing of Digital Circuit Designs. Proceedings of *the 21st Design Automation Conference* 1985, pp: 419-425

Kobus, R.: 2000, *Design Building Type Basics for Healthcare facilities: a building type basics handbook*, New York: Wiley

Laseau, P.: 1989, *Graphic Thinking for Architects and Designers*, New York, Van Nostrand Reinhold

Lee, G., Eastman, C. M. and Zimring, C.: 2003, Avoiding design errors: a case study of redesigning an architectural studio, in *Design Studies*, pp: 411-435

Nakakoji, K.:1993, *Increasing Shared Understanding of a Design Task between Designers and Design Environment: the Role of a Specification Component*, Ph.D. Dissertation, University of Colorado at Boulder

Nakakoji, K. and Sumner, T.:1994, Perspective-based Critiquing: Helping Designers Cope with Conflicts among Design Intentions, *Artificial Intelligence in Design* '94, Lausanne, Switzerland

(August), pp: 449-466

Nakakoji, K. and Yamamoto, Y. and Suzuki, T. and Takada, S. and Gross, M.:1998, Survey of Expert From Critiquing to Representational Talkback: Computer Support for Revealing Features in Design, in *Knowledge-Based Systems*, Vol.11, Issues 7-8, pp: 457-468

Repenning, A. and Sumner, T.:1992, Using Agentsheets to Create a Voice Dialog Design Environment, Proceedings of the 1992 *ACM/SIGAPP Symposium on Applied Computing*, pp: 1199-1207

Schön, D.:1985, *The Design Studio*, RIBA Publications, London

Schön, D. and Wiggins, G.: 1992, Kinds of Seeing and their Function in Designing, *Design Studies*, 13(2), pp: 135-156

Silverman, B.:1992, Survey of Expert Critiquing Systems: Practical and Theoretical Frontiers, *CACM (Communications of the ACM)*, Vol. 35, pp: 106-127

Solibri inc. : 2003 *http://www.solibri.com*, Solibri inc.

Stahovich, T., Davis, R. and Shrobe, H.:1996, Generating Multiple New Designs from a Sketch, in Proceedings of *American Association for Artificial Intelligent (AAAI)* '96, pp: 1022-1029

Stolze, M.: 1994, Visual Critiquing in Domain Oriented Design Environments: Showing the Right Thing at the Right Place, *Artificial Intelligence in Design* '94, Lausanne, Switzerland (August), pp: 467-482

Tversky, B: 1999, What Does Drawing Reveal About Thinking, in Proceedings of *the 1st Visual and Spatial reasoning in Design*, MIT, USA

Tversky, B: 2002, What do Sketches say about Thinking, in *2002 AAAI(American Association for Artificial Intelligence) Spring Symposium* Technical Report SS-02-08, pp:148-151

Woodbury, R., Burrow, A., Drogemuller, R. and Datta, S.:2000, Code Checking by Representation Comparison, in *Computer Aided Architecture Design Research In Asia*

(CAADRIA) 2000, pp: 235-244