DESIGN EVALUATOR

Critiquing Freehand Sketches

YEONJOO OH, MARK D. GROSS AND ELLEN YI-LUEN DO
Design Machine Group, University of Washington

Abstract. Design Evaluator is a freehand drawing environment that incorporates critiquing into a freehand sketch design system for early design. Design Evaluator's main features include a sketch interface, critiquing, and visualization of critiquing results. The system interprets the floor plan and provides critiques of three types when it finds conflicts between the sketched floor plan and built-in rules; text message, annotated drawing and 3D model/walk-through.

1. Introduction

1.1 BACKGROUND

It is a widely held view that designing is an iterative problem-solving activity. Some researchers have investigated the cognitive operations in the design process, or design cognition. For example, Schön's theory of reflection-in-action observes that designers generate a partially identified design solution, evaluate, reflect on, and then change it (Schön 1985). This reflection-in-action cycle continues until they are ready to go on to the next design problem. We propose that a design environment should provide feedback to help designers to find better solutions. The designer is not only generating design solutions. S/he keeps identifying the design problem through evaluating the proposed solution. In the design process, feedback about partially identified designs stimulates the reflection-in-action cycle.

1.2 MOTIVATION

Most CAD systems enable a designer to create external representations of designs, but they do not provide feedback. One possible method to give feedback during the design process is critiquing. Critiquing a proposed design offers the designer a chance of thinking again and improving it (Silverman 1992). In WEB-PHIDIAS (McCall, Holmes, Voeller and Johnson 1998), for example, a design collaboration system, all participants can author critiques about the proposed designs. The designers can think again and improve the design based on the critiques. Our project explores how a design support system can provide appropriate feedback.

Most critiquing systems and checker systems (e.g., code checking and floor plan checking) have their own structured drawing editors for interpreting the designs. Most critiquing system researchers invented custom drawing editors that work with their critiquing system. It is easy to capture the necessary knowledge from design proposals and to retrieve design critiques related with the design proposals (Hu et al. 2000). However these editors are not in general use. For example, if an architect designs a hospital and wants to get knowledge-based critiques about the design, the critiquing system cannot interpret the drawings. S/he must use the drawing editor of the critiquing system. However the drawing editors are so structured that they are limited as a design tool. These structured editors impose an additional cognitive load to the designer, so designers cannot explore and record their design ideas freely. Therefore, we are interested in incorporating critiquing in sketch-based early-stage design systems.

Designers usually prefer to sketch with pen and paper in the early stages of work. A sketch-based computer-aided design system is a tool to support the free exploration of ideas. Sketching activity bring designers to interact with their sketches which is examined in his mind during drawing. We want to build 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 designers can improve or revise their designs with the provided critiques.

The remainder of this paper is organized as follows. Section 2 describes related work in two categories: sketch-based design systems and critiquing systems. Section 3 describes the components of Design Evaluator – sketch interface, data structure, checkers and display manager, and modifying operations of the Design Evaluator. Section 4 concludes with a summary, discussion and directions for future research.

2. Related Work

2.1 SKETCH-BASED SYSTEMS

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. SketchVR (Do 2000) extended this work in the direction of creating 3-D models from 2-D sketches, which underlies the model display component of our current work in Design Evaluator.

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 two aspects of sketch-based systems: capturing knowledge from sketches and methods for representing designs.

SketchIT is a system for conceptual design of mechanisms. Although SketchIT (Stahovich 1996) is not a freehand drawing system, it identifies the behavior of the parts of a mechanism and derives constraints for making that behavior work. 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. For example, sKEA matches the rounded body of a cat and the rounded human torso. This matching capability can suggest what glyphs would be added and where they would be added, because usually people share graphic conventions and tend to sketch in the same way.

Other sketch-based systems support the designer with representing designs. Architectural design is concerned with shaping three dimensional space. For example, 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.

2.2 CRITIQUING SYSTEMS

A critiquing system is an effective way to use computer knowledge bases, because it provides feedback for designers to improve their design (Silverman 1992), yet minimizes the increase in the designer's cognitive load. It is because the critiques reduce the designer's information processing and they can make the designer concern about other design alternatives, based on the parts of the design that are pointed by the critiques, regardless of positive or negative critique.

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. 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.

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 critic 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 instead of text critiques.

Code checking and floor plan checking are long-standing problems in 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), floor plan checker (Kalay and Sequin 1995) and the commercial Solibri Model Checker (Solibri inc. 2003). However they have same view as 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.

3. Design Evaluator

The Design Evaluator program is a proof-of-concept working prototype built in Macintosh Common Lisp 4.3 to explore how to integrate critiquing into a sketch-based system. Design Evaluator includes a sketch-based design system and a critiquing system. It uses freehand sketching to input information about spatial arrangement in a floor plan. The inputted information is analyzed to give design critiques about spatial arrangement in the floor plan. It generates critiques when the system detects conflicts between the sketched floor plan and the rules embedded in its knowledge-base. Design Evaluator displays critiques in text and graphically highlights the problematic paths and spaces.

Design Evaluator has four components: sketch interface, data structure, checkers and display manager. Figure 1 shows the relations of these four components and the information flows between them. When a designer sketches the floor plan in the *Sketch Interface*, the sketched objects are stored in the *Data Structure*. Then the *Checkers* critique the sketched floor plan and generate critiques, if Checkers find problems in the floor plan. *Display Manager* displays the critiques.

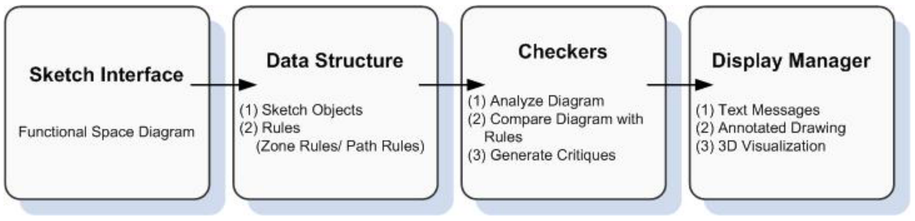
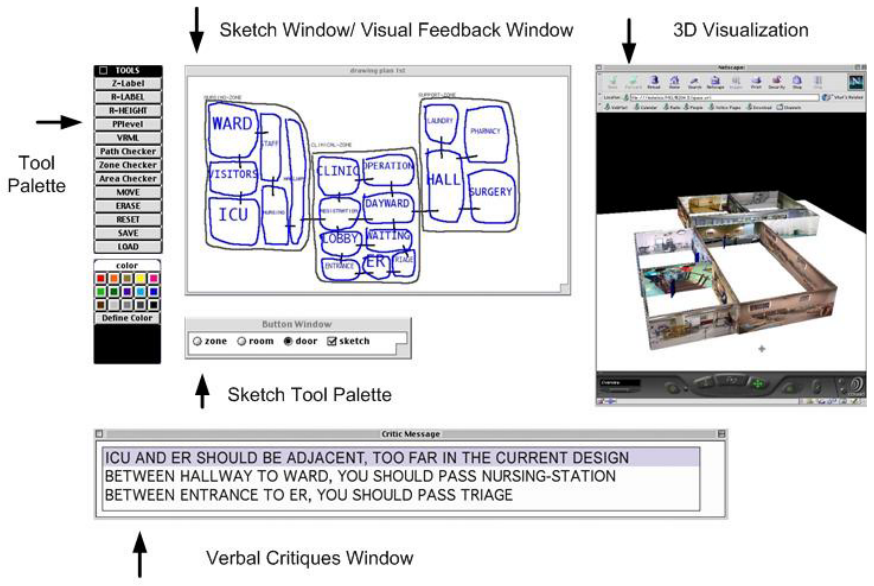


Figure 1: *Information Flow between Four Components of Design Evaluator*

Figure 2 shows the user interface of the Design Evaluator. The user interface is composed of three windows and two tool palettes. 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; graphical critique messages are also displayed here. A model/walk-through window shows the 3D model generated from the sketched floor plan. After a brief scenario showing how the system is used, the rest of this section describes these four components of Design Evaluator and their roles.

3.1 SCENARIO

Mike, an architect, is using the Design Evaluator to design a hospital. He starts by sketching the zoning based on a desired mass organization and program analysis. When he sketches rooms such as ward, clinic for outpatient and ER (emergency room) and doors, messages appear in the text critique window (see Figure 2). They tell Mike that *"BETWEEN ENTRANCE TO ER, YOU SHOULD PASS TRIAGE."* When Mike selects this text message, the Design Evaluator displays the problematic path on top of the sketch and highlights the wrongly placed rooms. The Design Evaluator tells Mike which path and room have a problem and what the problem is. With these text and graphic critiques, Mike revises the hospital design. To get a better view of the problem in 3D, and to see the problem from the hospital user's perspective Mike take a look at the 3D walk through model that the Design Evaluator has generated from the sketch. These models are texture-mapped with photos taken from rooms of different functional spaces in a hospital. Texture-mapped models give Mike a realistic and convincing simulation of the designed space. Mike goes on, revising the sketch to fix the problems.



ICU AND ER SHOULD BE ADJACENT, TOO FAR IN THE CURRENT DESIGN
BETWEEN HALLWAY TO WARD, YOU SHOULD PASS NURSING-STATION
BETWEEN ENTRANCE TO ER, YOU SHOULD PASS TRIAGE

Figure 2: *Design Evaluator User Interface*

3.2 DRAWING A FLOOR PLAN IN THE SKETCH INTERFACE

Design Evaluator is a freehand drawing environment. Designers use a stylus with a digitizing tablet to make freehand sketches of diagrams that indicate spatial arrangement of rooms in a floor plan. Users input two types of data: spatial boundaries and text labels. The designer draws the two kinds of bubbles and lines that represent the doorway connections. If the room is located in a certain zone bubble during drawing the room, the system records this relationship between room and zone. On the other hand, the line as doorway is laid across the rooms, the system stores this relationship in the data structure (see Section 3.3).

The sketch interface has two modes of display: sketch mode and rectified mode. The designer can choose display modes in the Sketch Tool Palette. If s/he checks a sketch box in the Sketch Tool Palette, the system displays sketch diagrams with the raw glyphs. The sketch interface of Design Evaluator takes rough drawings, recognizes the boundaries of sketched bubbles and converts them into straightened objects in the rectified mode. Users input the labels of rooms and zones by typing text directly on the drawing window. Figure 3 illustrates the process of converting sketched bubbles into a rectangles and labeling the drawing.

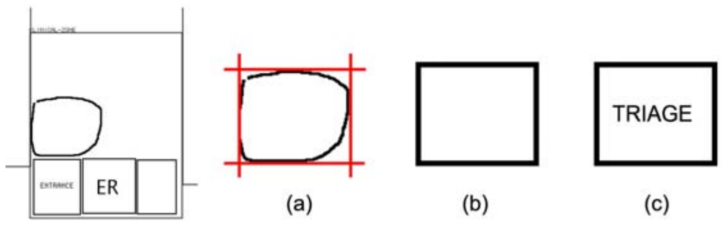


Figure 3: *Drawing Process in the sketch interface. Left figure is the part of the sketch diagram. (a) sketchy bubble diagram and recognizing the boundary of bubble; (b) rectified drawing after recognizing boundary of bubble; (c) labeling the functional names of zones and rooms*

3.3 DATA STRUCTURE

Design Evaluator is an object-oriented program. Its data structures store three kinds of information; sketched objects (rooms, zone boundaries, and doors between rooms), the analyzed results of the sketched floor plan (paths), and rules for offering critiques. Figure 4 shows the objects and their slots in the data structure.

To offer design critiques, the system needs domain knowledge. The current system knows two kinds of rules (path rules and zone rules) for checking the proposed spatial arrangement.

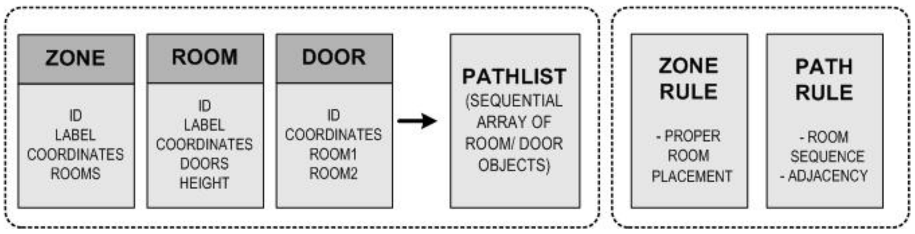


Figure 4: *Objects and Slots of Each Object*

These sketched objects' data structures are connected with each other. 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. 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 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**.

The system operates with two kinds of rules; zone rules and path rules. These rules describe many regulations of spatial arrangement. For example, one path rule is *(MUST-PASS-THROUGH ENTRANCE TRIAGE ER)*. It means that the patient must pass through Triage on the way from the Entrance to the ER (Emergency Room). An example of a zone rule is *(MUST-BE-IN CLINICAL-ZONE (ER TRIAGE CLINIC-FOR-OUTPATIENT DAYWARD…))*. It means that these rooms can and must be located in the CLINICAL-ZONE. Each rule is compared with the zone and room in the designer's sketch, and the paths that the path finder has derived. Other rules can be added easily. Checker compares the spatial arrangement of these objects and paths with the rules.

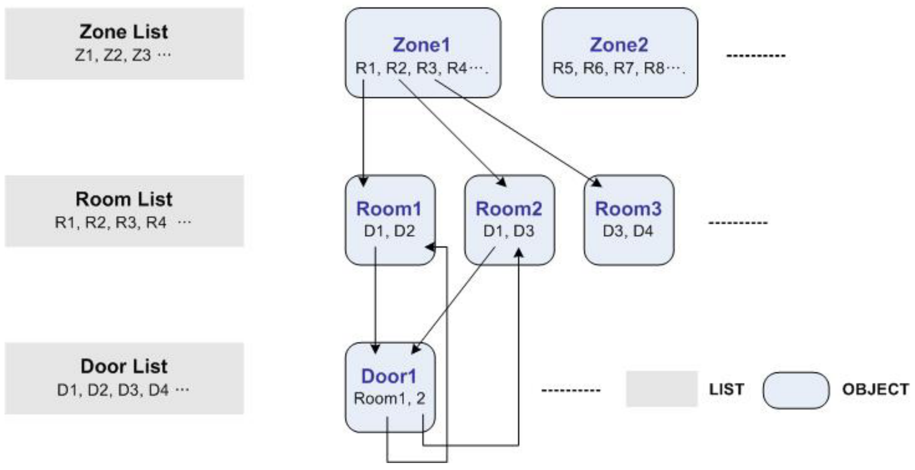


Figure 5: *Relations of Sketched Objects; Each zone has a list of rooms and each room has a list of doors. Each door knows which rooms it connects.*

3.4 CHECKERS

Rule Checkers perform three functions: analyze the sketched floor plan diagram, compare the sketched floor plan diagram with rules, and generate design critiques. Checkers compare the recognized spatial arrangement with the built-in rules. Then, if a Checker finds some errors, it generates design critiques, which are displayed by the Display Manager (see Section 3.5). Figure 6 shows the overview of the critiquing process.

3.4.1 Zone Checker

The Zone checker analyzes wrong or improper room placement in a zone. In the early design stage, an architect usually decides the program analysis with mass (zone) organizations. For example, hospital designs typically have three zones; Clinical zone, Nursing zone and Support zone. The designer decides the placements of rooms in each zone through program analysis. For example, ER (Emergency room) or ICU (Intensive Care Unit) should be in the clinical zone. Although these seem simple to decide, in a complicated building like a hospital, it is not uncommon to find incorrect placement of rooms. The Zone checker critiques the design: for each improperly placed room it identifies, it recommends the proper zone.

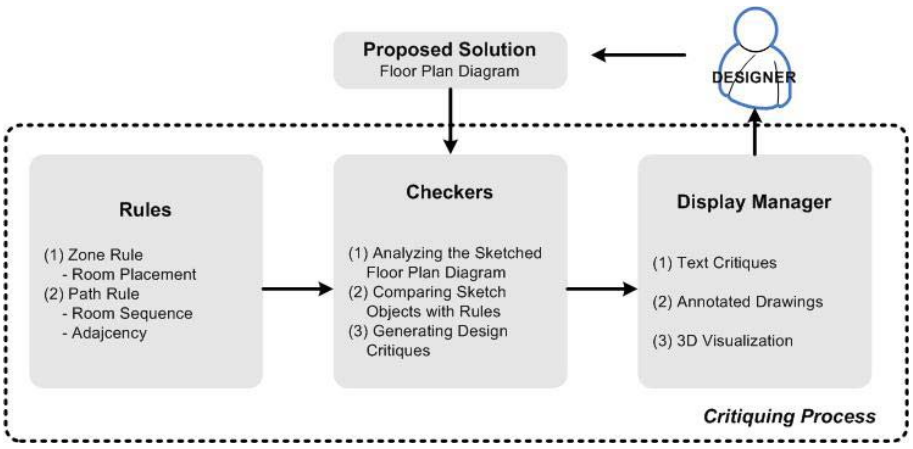


Figure 6: *Overview of Critiquing Process. (a) User proposed design solution with sketched floor plan; (b) Analyzing the sketched floor plan diagram; (c) Comparing the diagram with rules; (d) Generating design critiques, if there are errors or conflicts*

3.4.2 Path Checker

The Path checker identifies two kinds of problems with paths. The first is wrong or improper rooms in the path. Consider the path *(...- Hall – OR (Operation Room) – PACU (Post Anesthesia Care Unit) – Preoperative area – ICU – Ward ...)*. This path has an improper sequence of rooms: OR – PACU– Preoperative area. This path is bad because the path should follow the surgical process which happens in the following order: Preoperative area – OR – PACU. The Path Checker can point out this kind of problematic path.

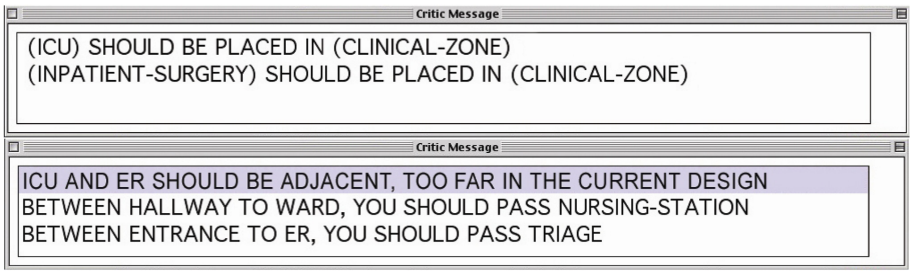
The second kind of problem is adjacency requirement. Consider the path *(ICU (Intensive Care Unit) – Nursing Station – Hallway – Ward – Inpatient Surgery – ER (Emergency Room))*. This path violates the adjacent requirement: ICU and ER. ICU and ER should be adjacent for effective medical treatment and patient delivery (Kobus 2000). The path is bad, because ICU and ER are too far apart. The Path Checker lets the designer know.

3.5 DISPLAY MANAGER

The display method is essential to show to the designer the problems the Checkers find. Here we describe three kinds of displays: as text, as annotated drawings and as an annotated 3D walk-through.

3.5.1 Text Critiques

If the two checkers find errors in the proposed design, the Display Manager generates text messages in the critique window. Figure 7 shows an example of text critiques. Each text critique is connected with a drawing annotation.



(ICU) SHOULD BE PLACED IN (CLINICAL-ZONE)
(INPATIENT-SURGERY) SHOULD BE PLACED IN (CLINICAL-ZONE)

ICU AND ER SHOULD BE ADJACENT, TOO FAR IN THE CURRENT DESIGN
BETWEEN HALLWAY TO WARD, YOU SHOULD PASS NURSING-STATION
BETWEEN ENTRANCE TO ER, YOU SHOULD PASS TRIAGE

Figure 7. *Text Critiques. (a) Zone Checker (b) Path Checker*

3.5.2 Annotated Drawings

The Display Manager annotates the design critiques on the designer's drawing. When the designer selects a message in the critique window, the Display Manager shows the problematic path and the poorly placed rooms in red as visual critiques. Figure 8 shows an example of the annotated drawing offered by Zone Checker and Path Checker.

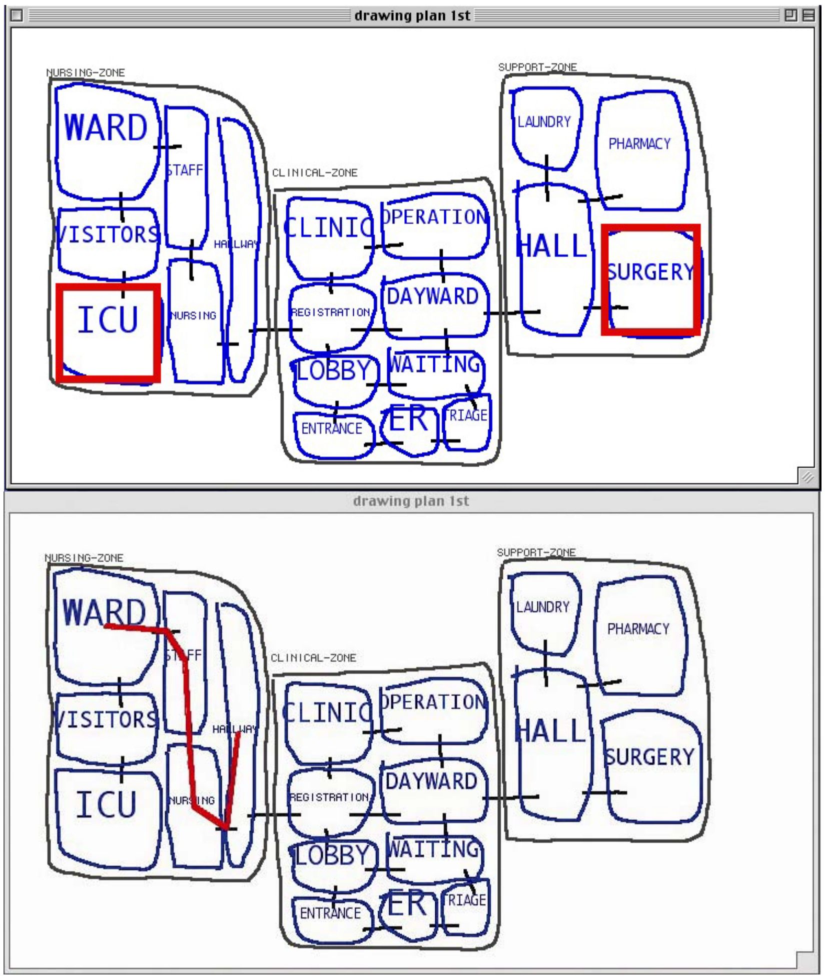


Figure 8: *Annotated Drawings. Display Manager displays the rooms and paths that have problems in red. (a)Zone Checker shows the wrongly placed rooms (b) Path Checker shows the problematic path between two rooms.*

3.5.3 3-D Visualization

The third display method is a three dimensional model of the floor plan with VRML (Virtual Reality Modeling Language). This display has two merits. First it shows the spatial arrangement and relations of the designed rooms in a three dimensional model. Second, texture-mapped models give the designer a realistic and convincing simulation of the designed space. This 3D visualization helps the designer to detect problematic room arrangement and help her to locate himself in the 3D visualization of the floor plan she has designed. Figure 9 shows the texture-mapped VRML models in the web browser. A walk-through 3D model can deliver information about problems to designers more effectively.

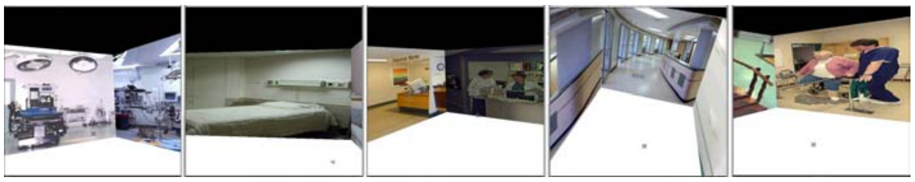


Figure 9: *3D Texture-mapped 3D VRML Models: Operating Room, Ward, Nursing Station, Hallway and Physical Therapy Room*

3.6 REPAIR THE FLOOR PLAN DIAGRAM (EDITING OPERATIONS)

When the designer gets feedback from the Design Evaluator, the designer might change the current floor plan diagram. 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. Therefore, the system recomputes the relations of objects and updates all the data.

3.6.1 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. For example, if the system removes a room object specifically, it also must remove the room from all the doors that currently refer to it. If the designer wants to remove the room, the erase function must perform a series of removing activities: (1) removing the room from the Room List, (2) Erasing the room slot from its zone, and (3) Erasing door objects.

3.6.2 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 new taken position specified by a second click. Figure 10 describes a series of activities of the move function: (1) calculating the new coordinate position with the new taken point, (2) recomputing which zone the room is placed in, and (3) removing the door objects that are related to the moved room.

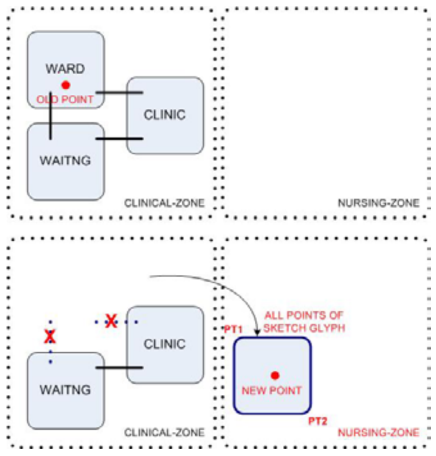


Figure 10. *A Series of Activities for Moving Room*

4. Discussion

We have shown how critiquing can be integrated into the sketch interface in early design. Unlike other critiquing systems which employ structured drawing editor, Design Evaluator incorporates critiquing in a sketch-based design system.

We gain two advantages from incorporating critiquing in a sketch-based system. One is to improve sketch design system. Through sketches the designer interacts with mental imagery, analyzing and examining design ideas. This mental imagery from the sketches triggers generation of new design solutions. The offered critiques could stimulate mental imagery from sketches.

Design Evaluator is embedded with rules about spatial arrangement. When conflicts happen between sketched floor plan and the built-in rules, Design Evaluator lets the designer know these conflicts. This gives the designer a chance to improve and to revise the design solution and eliminates problems that might otherwise remain undetected.

Design Evaluator is based on a sketch-based design system. When the designer becomes aware of errors in the design from critiques, he/she would to want revise it. Sketching makes design revision easy and allows the designer to explore other design ideas quickly.

Design Evaluator delivers design critiques in three ways: text critiques, annotated drawing, and 3-D annotated walk-through. Design Evaluator is intended for architects, who communicate their designs with themselves and other people using text and graphical modes. They can have trouble to communicate their designs with only one mode, displaying graphical critiques with text critiques is more effective to the designer.

Design Evaluator supports sketching design providing design critiques. When the designer receives the critiques, designer’s next activity is to evaluate the design and revise it. Sketch-based system makes the revision of design easy and it allow designer to explore ideas continuously. Design Evaluator environment supports designer’s reflection-in-action cycle effectively, because the offered critiques stimulate design cycle.

5. Future Work

Future work at this point includes adding other checking features, making interface for changing the rules according the designers and design task, improving the usability of the sketch-based system and expanding Design Evaluator with an Advisor component.

First, several other checking features can be added into Design Evaluator. These include circulation conflicts, the capability of calculating dimensions of sketched objects for ADA compliance, and the support of several floor plans for checking vertical circulation.

Second, we plan to strengthen the Design Evaluator’s sketch to support the design process freely. We plan to strengthen the Design Evaluator’s sketch recognition to include high-level object relationships such as “overlapping”, or “next to”. In addition, we plan to make the Design Evaluator recognize multiple level floor plans for vertical circulation checking rules and vertical zoning. 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. Therefore the sketch interface should be improved with other sketch supports in order to allow exploring design ideas more freely.

The Design Evaluator has embedded rules for analyzing the floor plan. Design tasks are not always in same situation. For supporting the various design tasks, if Design Evaluator allows the designer to create or to change the rules for checking the designs, it would be more powerful. On the other hand, if the system can load checking modules for other building types or other design domain, it would be more genenal.

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 position 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.

The designer receives only critiques from the Design Evaluator. Design Evaluator could also give suggestions to designers besides the critiques, if an "advisor" component is added. The advisor component would show how to fix the specified problems, for example, by offering examples from relevant architectural precedents.

Acknowledgements

This research was supported in part by the National Science Foundation under Grant CCLI-EMD DUE-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.

References

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

Gross, M.: 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. and Do, E.: 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

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 Mechnical Engineers (ASME) Design Engineering Technical Conferences 2000*

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

Kalay, Y. and Sequin, C.:1995, 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. et al.: 2000, *Design Building Type Basics for Healthcare facilities: a building type basics handbook*, New York: Wiley

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 *in 1992 ACM/SIGAPP Symposium on Applied Computing*, pp: 1199-1207

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

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

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