THE THREE R'S OF DRAWING AND DESIGN COMPUTATION

a drawing centered view of design process

MARK D GROSS, ELLEN YI-LUEN DO
Design Machine Group, Department of Architecture, University of Washington, Seattle, WA 98195-5720, USA

Abstract. A drawing centered view of design process focuses on the interplay between designer expertise, domain knowledge and media manipulation. We report on computational sketching software systems we have implemented to support recording, reasoning, and resolving in design.

1. Introduction

1.1. MOTIVATION AND RELATED WORK

Ivan Sutherland’s Sketchpad system first demonstrated the power and promise of interactive, intelligent, pen based graphics to support engineering design (Sutherland 1963). Among other innovative ideas, Sketchpad employed what later came to be known as object-oriented programming, graphics scene-graph hierarchy, interaction with a stylus, on-screen menus, two-handed interaction, and a relaxation-based constraint solver to maintain user-specified design constraints on a drawing. In addition to these technological innovations, Sketchpad demonstrated how computers might support semantically attached interactive design drawing, yet for many years little further work was done.

Several factors fueled a resurgence of interest in pen-based computing, beginning in the 1990s. Low cost LCD screens and tablet-stylus technology led to pen-based PDAs such as the Apple Newton, Palm Pilot, and others, as well as early pen computers and recently the Tablet PC. This new hardware combined with advances in pattern recognition rekindled interest in drawing-based systems for design. Over the past few years a great deal of new work has appeared on sketch design systems.

Many have noted the affordances of sketching in design. Sketching, for example, allows quick exploration of ideas at a high level of abstraction, avoids early commitment to a particular solution, allowing many alternatives to be explored (Fish and Scrivener 1990; Ullman, Wood et al. 1990). Some experienced designers postpone using computer-aided design tools because they feel that these tools require a degree of precision and commitment that is inappropriate for the early phases of design. They prefer to develop conceptual design sketches with pencil and paper, moving to computer-aided tools at a later stage of designing (Cross, Christiaans et al. 1996; Suwa and Tversky 1997).

Many systems aim to understand drawing to provide design feedback. For example, SILK enables a designer to sketch and test a user interface design prototype (Landay 2001). EsQUIsE interprets architectural sketches for design performance evaluation (Leclercq and Juchmes 2002). An increasing number of researchers consider sketch understanding a knowledge-based task. They argue that computer can provide domain knowledge for the task through sketching. For example, the recognition of device symbols and connections among them in a motor driven conveyor system diagram could infer motion by reasoning from domain knowledge (Kurtoglu and Stahovich 2002). The ASSIST mechanical engineering sketch design system can simulate mechanical movements and create movies of them, such as a car rolling down a hill (Davis 2002). Forbus et al, working on military "course of action" diagrams minimize the role of sketch recognition. They argue that sketch recognition is difficult for computers and easy for people, and the same information can be obtained from other sources such as speech and deictic references (Forbus, 2001). Their system therefore attempts only basic recognition of spatial pointers (arrows) combined with speech recognition. Their structure-mapping engine (Falkenhainer, Forbus et al. 1990) is used to infer and interpret user actions.

There is also a growing recognition of the value of ambiguity and tolerance for multiple interpretations (Mankoff, Hudson et al. 2000; Gaver, Beaver et al. 2003). Saund's "perceptual sketching" systems (Saund and Moran 1994; Saund, Fleet et al. 2003) identify patterns in the lines of a sketch that enable users to select and work with the (emergent) objects that they see—regardless of how the sketch was initially constructed. Oviatt & Cohen's QuickSet system (Oviatt and Cohen 2000) combined sketch and speech recognition, showing that cues from sketch recognition can improve speech recognition performance, and vice-versa. Other work on computational support for drawing focuses on using sketches to produce more refined three-dimensional models (Igarashi and Hughes 2001; Contero González, Naya Sanchis et al. 2003).

1.2. DESIGN: EXPERTISE, KNOWLEDGE, AND MEDIA

Observing designers at work, we find them simultaneously engaged in three rather different kinds of tasks. At a practical level we see that they work with various media, for example making drawings and models and using these media in communicating with others about the design. We see also that they engage domain knowledge, for example making predictions and exercising judgments about how a design will perform. And we see designers allocating their time to different tasks in the design process, at some points searching for information, at other points generating new ideas, testing and evaluating alternatives, communicating with colleagues and clients, or developing basic concepts into specific detailed proposals. Hence we propose to look at design in terms of three distinct capacities: as the manipulation of different *media* with *domain knowledge*, governed by *design expertise* (figure 1).

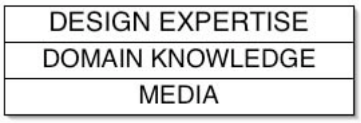


Figure 1. Levels of computational support for design.

This understanding implies three fundamental components of computational support for design. First, computational tools must support the media that designers use to create, edit, view and review, and exchange designs and design ideas. These media include drawings and text as well as speech, audio, video, gesture, and collage. Designers are trained in the use of media to create, capture, consider, and convey ideas. Although conventional computer media for design, i.e., CAD drafting and modeling, has primarily focused on hard-line drawings and three-dimensional geometry, a great deal of creative work in design is actually carried out in the medium of freehand sketches and diagrams.

Second, computational tools must integrate domain knowledge into the design decision-making process. Designers know a lot about their domains, and they exercise a great deal of intelligence in making design decisions. On occasion, they refer to handbooks, case studies or well-known examples, previous similar projects; they call for technical analyses of projected design performance; they seek expert advice. Computational support for domain knowledge in design has been weighted heavily toward end-stage analysis and evaluation based on detailed design representations constructed with structured editors, rather than sketches and freehand drawings.

Third, computational tools must embody expertise in managing the design process. That is, independent of any specific design domain, the tools must help a designer manage the design process — ideation and retrieval of relevant precedents, generation and evaluation of alternatives, consulting experts, balancing stakeholder values, etc. Our work so far has concentrated on the relationship of design media and domain knowledge, and most of the work outlined here is in this territory. We return to consider the question of design expertise in the final section of this paper.

1.3. DRAWING—THE CENTRAL REPRESENTATION IN DESIGN

We take a drawing-centered view of designing. In domains such as mechanical engineering and architectural design where the product is a physical object, the drawing is typically the single representation that the designer uses throughout the designing process, from initial rough sketch to final fabrication drawing. Although other representations (such as specifications, component lists, and schedules) also play roles, the drawing remains the focus of design activity.

Our view of design is colored in the first place and most strongly by our own training and experience as practicing designers and by first-hand observation of colleagues and students in the architecture schools where we have worked, as well as in other disciplines (such as mechanical engineering) where design is taught. Although we have done a few studies of design activity (Do, Gross et al. 1999) we largely rely on empirical work by Goldschmidt, Tversky, and others for insights about the cognitive roles and functions of drawing in design (Goldschmidt 1994), (Schon 1992; Tversky 2002).

1.4. THREE R'S OF DRAWING AND DESIGN

We outline in Figure 2 various activities that designers carry out using drawing as a base representation. We group these activities into three categories (the three R's of design): recording, reasoning, and resolving. **Recording** activities concern design and knowledge capture—from the user's pen input at the stroke level to capturing speech and text during designing, including design rationale and other annotations that play a role in collaboration and negotiation. **Reasoning** activities engage domain knowledge in various ways—deducing the performance behavior of a proposed design, retrieving relevant design precedents or cases from a library or database, stimulating creative thought through reasoning-by-analogy, and so on. The third R, **Resolving**, concerns the development of drawings over the design process from rough, sketchy, and abstract representations to specific, definite, and well-defined design proposals.

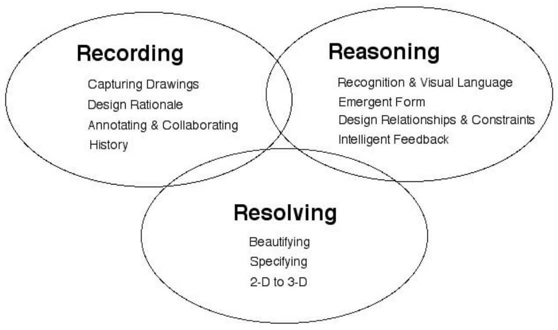


Figure 2. Computational sketching systems must support diverse design activities.

1.5. OUR MULTIFARIOUS EFFORTS TO SUPPORT DESIGN DRAWING

Over the past decade we have engaged in a research program and developed computational support to investigate various aspects of designing. We began with a recognition-based sketching program written in Lisp called the Electronic Cocktail Napkin, which formed the platform for several initial forays into the territory. These efforts focused principally on how to recognize and interpret freehand drawings and extract and embed semantic content. We later extended this work with several components focused on other aspects of designing. For the most part, we developed each module separately, following the exigencies of student interest and funding. The result is a collection of apparently diverse projects that are published in disparate venues. Yet taken together, we believe that they outline the principal components of a drawing-centered design system. This proof-of-concept software illustrates how many of these parts could work. The purpose of the current paper is to assemble this body of work in relation to our view of design and decision support. In the interest of providing a larger picture, we omit technical details, which can be found in the reports we cite on the individual projects. This paper provides a reflecting commentary and framework to relate how all these software systems in the context of drawing and design computation support.

We feel compelled to point out that although most of the examples in the work described below are from architecture, the model of designing and the systems we have built are quite independent of this domain. Although we would be pleased to see computer support systems for architectural design built along the lines we outline, we feel the strategies presented here could be incorporated into a wide variety of design support systems.

2. Drawing Tools to Support the Three R's of Design Activities

2.1. RECORDING – KNOWLEDGE CAPTURE

The capture and management of drawings is central, but important additional information about a design often accompanies drawings in various media—speech, text, photographs, even physical samples of materials—and at various levels of formal representation.

Recording activities concern design and knowledge capture. At the most basic level we record the user's pen input strokes, pressure and speed using the Electronic Cockail Napkin as a platform for diagram parsing and understanding. The Design Amanuensis and Design Recorder projects extended the capture to include speech and text and provide synchronized history for reply and search. To support shared drawing, annotation of design rationale, collaboration and negotiation, we developed the Immersive Redliner and Space Pen projects which support design drawing in 3-D and also the synchronous shared whiteboard drawing with design history in NetDraw. Below we briefly describe each project and their roles in Design Recording as illustrated in Figure 3.

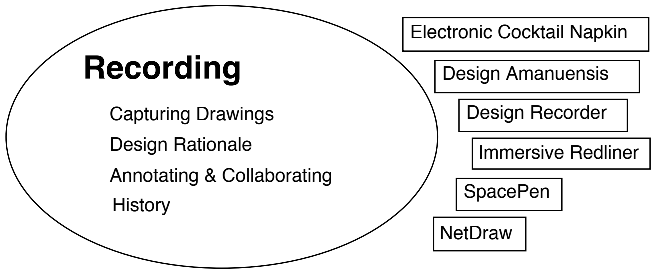


Figure 3. Recording activities concern design and knowledge capture.

2.1.1. Capturing strokes and recognizing drawing elements

The Electronic Cocktail Napkin (Gross 1996) captures the designer's drawing strokes from a tablet. The program stores raw data as a time-stamped sequence of tuples (x, y, and pressure) which it then analyzes to extract features such as drawing speed, corners and points of inflection, direction, bounding box, size, and aspect ratio, and path through a 3x3 grid inscribed in the bounding box. These features form the basis of a matching scheme in which glyphs drawn on the tablet are compared against a library of previously trained templates. The recognizer returns a set of most likely matching glyphs along with certainty values and details about the match (for example, that a figure was recognized but drawn upside down). If the program cannot identify a figure, recognition is simply deferred. Figure 4 shows examples of user-trained symbols and configurations (assemblies of symbols) the system recognizes.

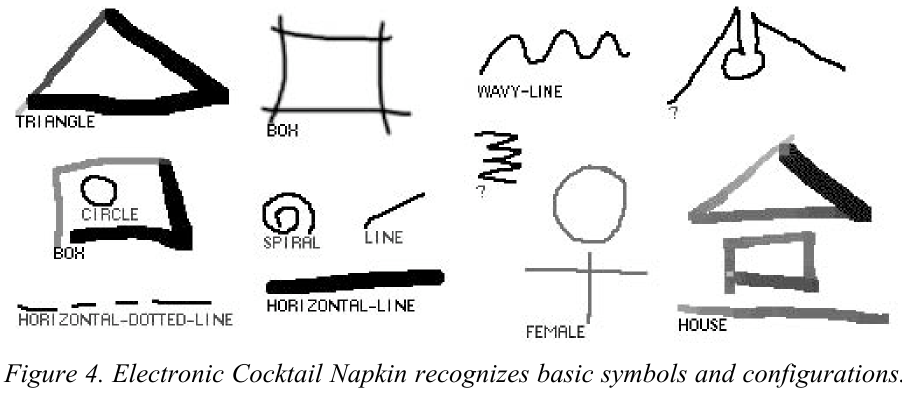


Figure 4. Electronic Cocktail Napkin recognizes basic symbols and configurations.

2.1.2. Recording multi-modal design conversations

The Design Amanuensis combines the Napkin's stroke-capture with a continuous speech-to-text recognizer, storing the multimodal design conversation for later random-access playback and analysis (Gross, Do et al. 2001). The converted speech record enables search for key phrases ("where in the designing did we talk about 'support column size?' "). The search not only finds items in the spoken and text record, but it also highlights the drawing activity that took place at the time. The Napkin's graphical search facility also enables search for particular figures—which similarly are linked through their timestamps to the associated text and speech component of the design record. Figure 5 shows the drawing, control panel, and timeline views of the design record; the user has selected text, which highlights the associated graphics.

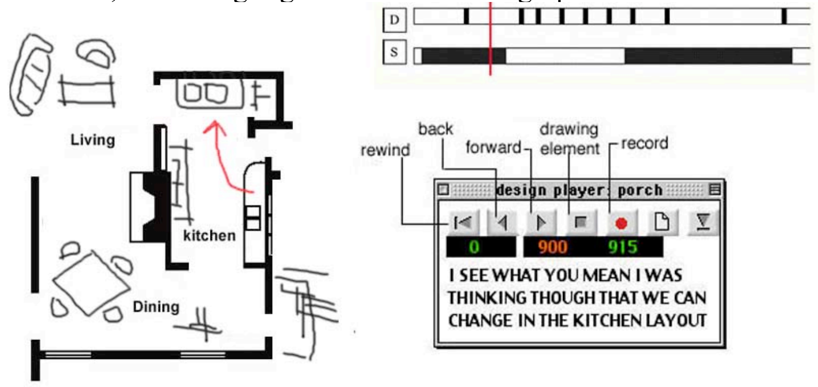


Figure 5. Design Amanuensis offers recording, playback, and search of the graphical and spoken design conversation.

The Design Recorder is a later implementation that uses the Tablet PC to capture speech and drawing input from distributed collaborating designers. Figure 6 shows the chunking of speech and text, displayed as selectable icons along a timeline.

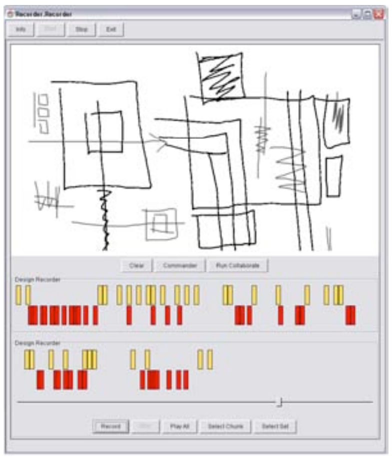
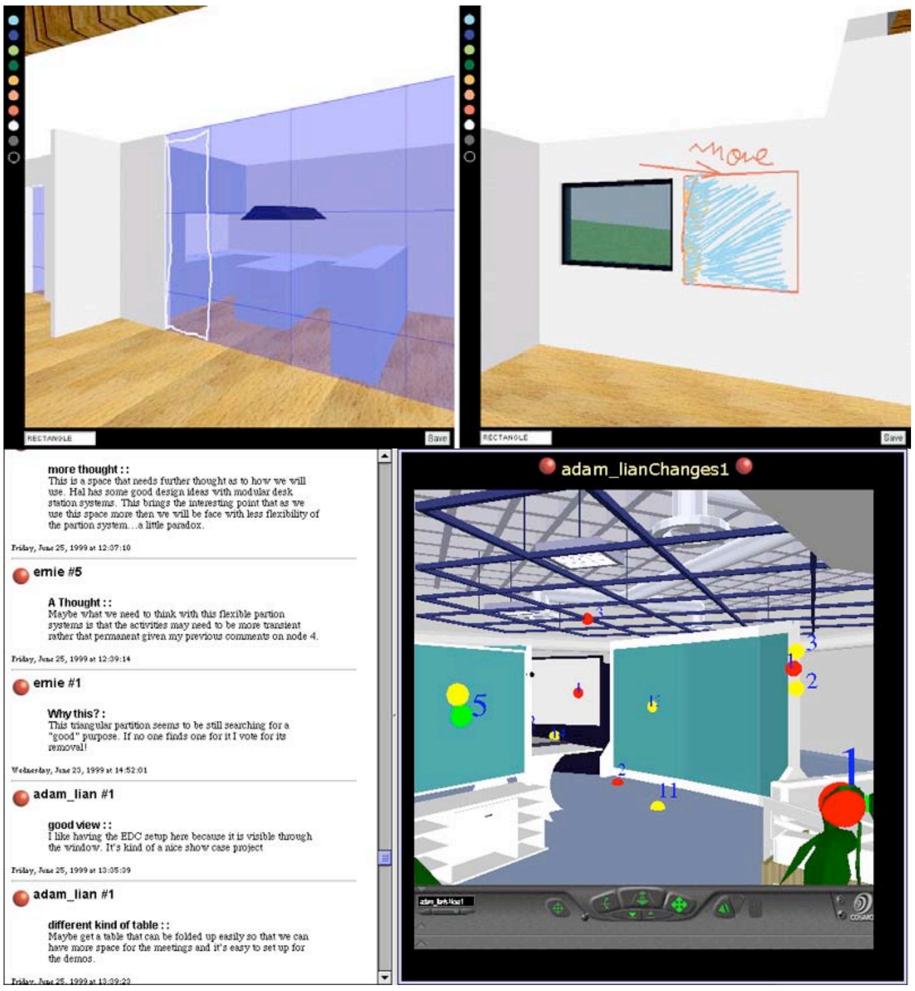


Figure 6. Design Recorder: above: design drawing of two collaborating designers; below: 'chunks' of speech and drawing.

2.1.3. Recording and embedding rationale in designs

The SpacePen (and a predecessor project, the Immersive Redliner) explored design annotation in three-dimensional models (figure 7). Collaborating designers (or other stakeholders) sketch and attach text annotations to mark up a shared three dimensional model (Jung, Do et al. 2002). For example, a designer can propose adding a window by drawing directly on an existing wall in the 3-D model (figure 7 top). The system performs simple sketch recognition to identify symbols such as arrows and rectangles. Sketched lines and text annotations appear to subsequent viewers of the model and the designer can later act on the proposal. The system stores design critiques and rationale expressed as drawings or text which are located in the spatial context where they apply.



more thought : :
This is a space that needs further thought as to how we will use. Hal has some good design ideas with modular desk station systems. This brings the interesting point that as we use this space more then we will be face with less flexibility of the partion system…a little paradox.

Friday, June 25, 1999 at 12:37:10

ernie #5

A Thought : :
Maybe what we need to think with this flexible partion systems is that the activities may need to be more transient rather that permanent given my previous comments on node 4.

Friday, June 25, 1999 at 12:39:14

ernie #1

Why this? :
This triangular partition seems to be still searching for a "good" purpose. If no one finds one for it I vote for its removal!

Wednesday, June 23, 1999 at 14:52:01

adam_lian #1

good view : :
I like having the EDC setup here because it is visible through the window. It's kind of a nice show case project

Friday, June 25, 1999 at 13:35:39

adam_lian #1

different kind of table : :
Maybe get a table that can be folded up easily so that we can have more space for the meetings and it's easy to set up for the demos.

Friday, June 25, 1999 at 13:39:23

Figure 7. Recording design rationale and sketch annotations in a 3-D model – Immersive Redliner (below) and SpacePen (above).

2.1.4. Design History

NetDraw (figure 8) offers a shared drawing surface that enables several designers to work together simultaneously (Qian and Gross 1999). We also used NetDraw to explore several ideas related to synchronous collaboration. NetDraw stores and displays a history of the designing, so that a designer can return to a previous state and proceed to design from that point. NetDraw also offers an *ephemeral gesture* feature whereby a designer can sketch temporary marks over the design. These marks, useful for deictic references ("here is the main circulation path through the building") appear momentarily on other designers' drawings, then gradually fade away.

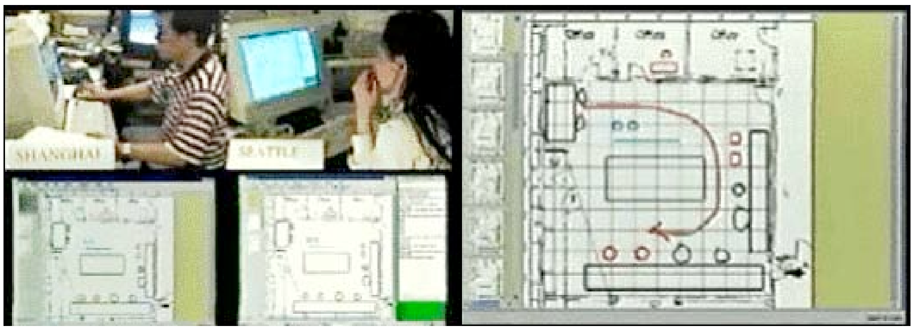


Figure 8. NetDraw: design history, ephemeral arrow gesture for deictic reference.

2.2. REASONING – DOMAIN SEMANTICS WITH INTELLIGENT SYSTEMS

The work described in the previous section treats drawings and attendant information simply as graphical data—one might say, as informal (human-readable) design representations. But much of our interest in drawing as a design representation hinges on the knowledge that designers embed in drawings and the reasoning they apply to work on and with drawings. That is, we are interested in the intelligent processes of which design drawing is a part, and the ways in which we can use knowledge based computational systems to enhance design drawing.

Design drawings embed the designer's understanding of a domain and assumptions and decisions about the design; and a computer program can use recognition processes to extract this information, and then reason about it.

Reasoning activities in design engage domain knowledge in various ways–including the performance behavior of a proposed design, retrieving relevant design precedents or cases from a library or database, stimulating creative thought through reasoning-by-analogy, and so on. We have implemented several software systems to support design reasoning in the form of connecting intelligent systems of domain knowledge to provide design feedback (figure 9). The Napkin's visual language parsers identify the context of drawing elements and configuration. It also supports recognition of perceptual figures such as emergent shapes from overlapping figures. The Stretch-a-Sketch program maintains interactive behavior among drawing elements using constraint propagation. IsoVist, Design Evaluator and Napkin-Archie programs support intelligent feedback such as performance analysis, critiquing and retrieval of relevant cases from a database. Light Pen offers design decision support guided by lighting design practice rules. LAN-Tools activates a simulation of network communication and proposes modifications based on designer's hand drawn diagrams.

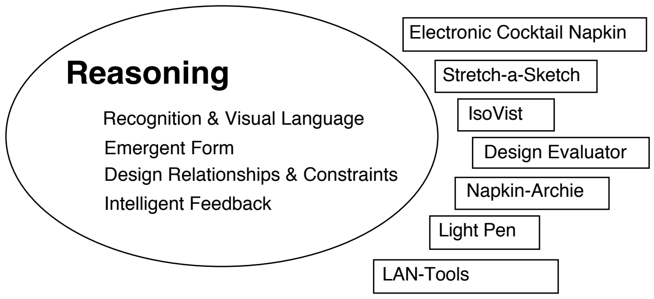


Figure 9. Reasoning activities engage domain knowledge.

2.2.1 Recognition and Visual Language, Emergence

If computers are to support design drawing as a knowledge-based activity, then they must recognize and interpret drawing semantics. The heart of the Electronic Cocktail Napkin is the symbol recognizer described earlier combined with a visual language parser that identifies complex configurations built up hierarchically from simpler components. For example, the system recognizes a building façade as composed of windows, doors, and a roof composed in certain spatial relationships. The visual language allows for multiple alternative parses, so that a drawing can be “read” in several different ways, variously grouping its components into alternative assembly graphs.

The Napkin’s recognition scheme is contextual: depending on the drawing context the program recognizes symbols and configurations differently (Gross and Do 1996). For example, the same symbol may be recognized in an analog circuit diagram as an inductance and in a mechanical drawing as a spring. Conversely, when the Napkin identifies a symbol or configuration that is unique to a context (for example, a transistor symbol in an analog circuit diagram) then it uses this to determine the context and thereby resolve pending recognition ambiguities.

A first task is recognizing the drawing elements that the designer intended to represent—walls, columns, and windows in a floor plan, for instance, or levers, pulleys, and gears in a mechanical drawing. However, drawings often contain unintended figures formed by spatial relations among intentionally drawn components, termed emergent shapes or forms. These shapes stand out to the perceptive human designer, and an intelligent drawing system must identify them also. We therefore developed a component of the Electronic Cocktail Napkin that searches the designer’s drawing for symbols that emerge by combining strokes from two or more spatially proximate symbols and sub-strokes formed by intersecting and connecting strokes (Gross 2001). The system first generates a large set of candidates; then the symbol recognizer selects those that match previously stored templates. For example, if the designer draws a diamond inscribed in a square, the program, which has previously been trained to recognize triangles, identifies the four corner triangles (Figure 10).

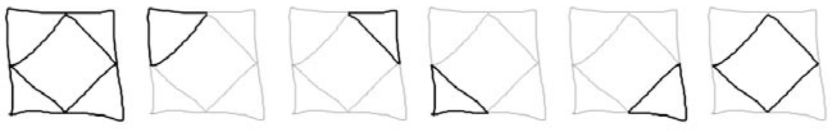


Figure 10. Triangles and diamonds identified in the first (leftmost) diagram.

2.2.2 Constraints bring drawings to life

A designer sees in a drawing more than a static arrangement of arbitrary symbols; s/he sees its potential transformations. Domain semantics circumscribe the syntactic transformations of the diagram that are considered legal. For example, in transforming or editing a diagram of a mechanism—or a molecule—the designer maintains its essential spatial relationships. Which relationships in the diagram are essential and which are arbitrary depends on the domain. In an architectural plan geometry is essential; in an analog circuit diagram, it is not.

The Stretch-A-Sketch system (figure 11) uses a constraint propagation engine to augment a drawing with interactive edit behavior (Gross 1994). First the system extracts the spatial relationships from the diagram—identifying, for example, elements that are connected, contained, adjacent, and so on. Then it instantiates these found relationships as constraints on the diagram elements. Subsequently as the designer edits the drawing, the system then maintains these constraints, so that connected, contained, adjacent, elements stay connected, contained, adjacent etc.

Stretch-A-Sketch simply extracts the constraint relationships from the drawing. A more sophisticated version would use domain knowledge—supplied by a module external to the sketching system—to select the appropriate relationships to apply to the drawing.

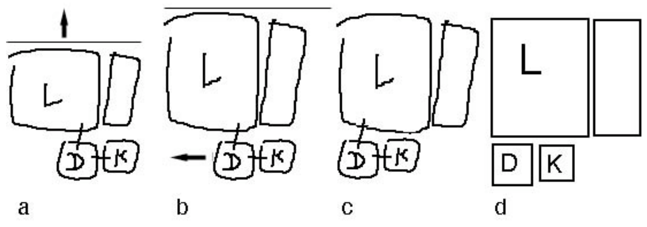


Figure 11. Stretch-a-Sketch interprets a floor plan (a) It infers an align-top constraint so that (b) stretching L also stretches the adjacent room, (c) rooms D and K retain their adjacency as D moves left, (d) the diagram rectified.

2.2.3 The sketch as locus for intelligent feedback

In Stretch-A-Sketch, domain knowledge adds behavior to the designer's drawing through constraints. In this spirit, two other projects explore the idea of using the sketch itself as the locus of domain feedback. The IsoVist program (figure 12) displays the result of design performance analysis overlaid on the designer's sketch (Do and Gross 1997). The program calculates the area of a floor plan that is visible from a given vantage point (an "isovist"), and displays this polygon on the floor plan sketch. The designer can move the vantage point around the floor plan to obtain an isovist calculation at various locations.

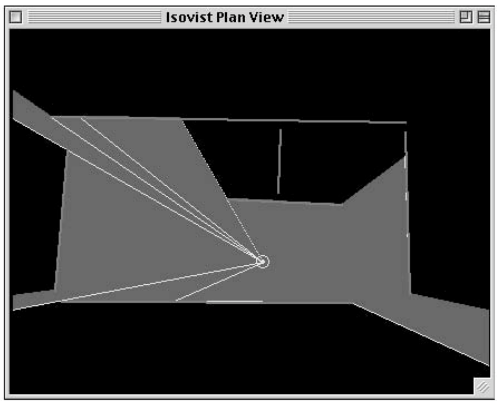
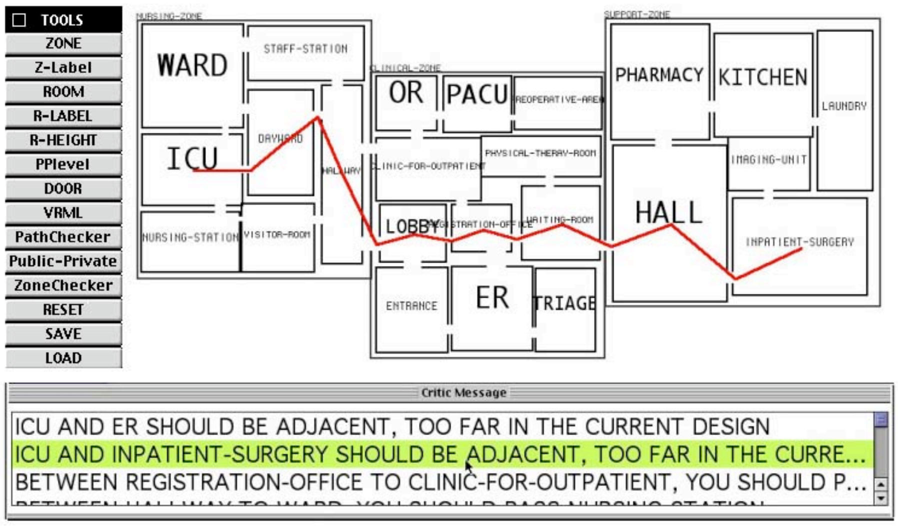


Figure 12. IsoVist overlays design performance evaluation on the sketch.

Another example of displaying the results of domain based design performance evaluation is the Design Evaluator (figure 13), a system for critiquing sketched designs, which analyzes a floor plan layout for spatial arrangement and circulation problems (Oh, Gross et al. 2004). When the system's rule checkers ('critics') detect problematic conditions, the Design Evaluator notifies the designer using text messages that are linked to a graphic overlay that marks the problem on the plan. For example, if the system finds a problem with a circulation path, it will highlight the path on the floor plan drawing.



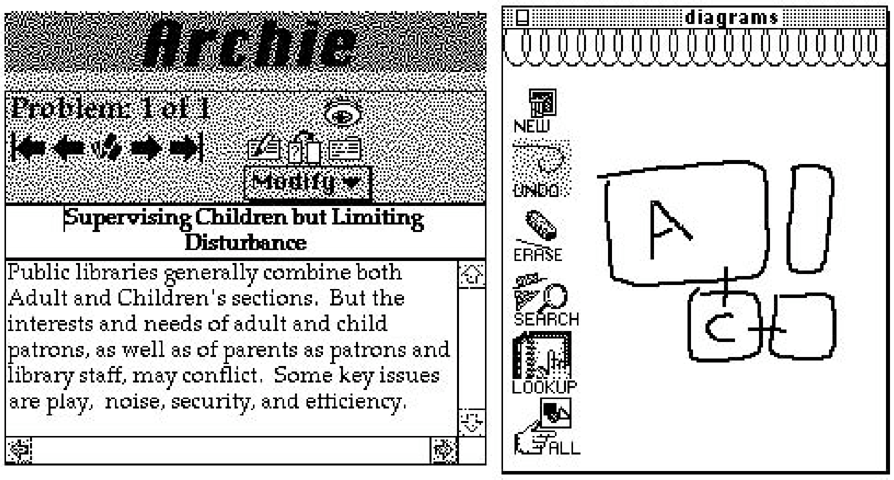
ICU AND ER SHOULD BE ADJACENT, TOO FAR IN THE CURRENT DESIGN
ICU AND INPATIENT-SURGERY SHOULD BE ADJACENT, TOO FAR IN THE CURRE...
BETWEEN REGISTRATION-OFFICE TO CLINIC-FOR-OUTPATIENT, YOU SHOULD P...
BETWEEN HALLWAY TO WARD, YOU SHOULD PASS NURSING STATION

Figure 13. Design Evaluator highlights the designer's drawing (originally sketched, now rendered in rectified form) with graphical critiquing. Here the path critic points out a problematic path in the floor plan.

2.2.4 External intelligent systems

A design drawing can serve as input to various external design aids or tools, which can diagnose problems and opportunities in the design, suggest relevant or interesting references, offer advice, or evaluate design performance.

The Napkin-Archie system (figure 14) links the Cocktail Napkin with Archie, a case base of building designs with associated stories, problems, and solutions (Gross, Zimring et al. 1994). We added a diagram index to the Archie case base, enabling the designer to retrieve design cases that match a given drawing. For example, Napkin-Archie might recognize a problematic entrance condition in the drawing and retrieve a similar design case from the Archie library, illustrating the problem and potential solutions.


Archie

Problem: 1 of 1

Modify

Supervising Children but Limiting Disturbance

Public libraries generally combine both Adult and Children's sections. But the interests and needs of adult and child patrons, as well as of parents as patrons and library staff, may conflict. Some key issues are play, noise, security, and efficiency.

Figure 14. Napkin-Archie indexes a case based design aid with diagrams.

A similar scheme drives our "shape based reminding" system, in which drawing is used to retrieve visual images from a database (Gross and Do 1995). Each stored image is indexed with a diagram. Given a source drawing, the system uses a multi-dimensional matching technique that compares the numbers, shapes, sizes, and spatial relationships of elements to retrieve visually similar images. Depending on the image collection, the system can be used within a domain (e.g., to find components in a catalog that most closely match the designer's drawing) or across domains, as a creativity-stimulating scheme (e.g., using a floor plan diagram to search a database of flowers).

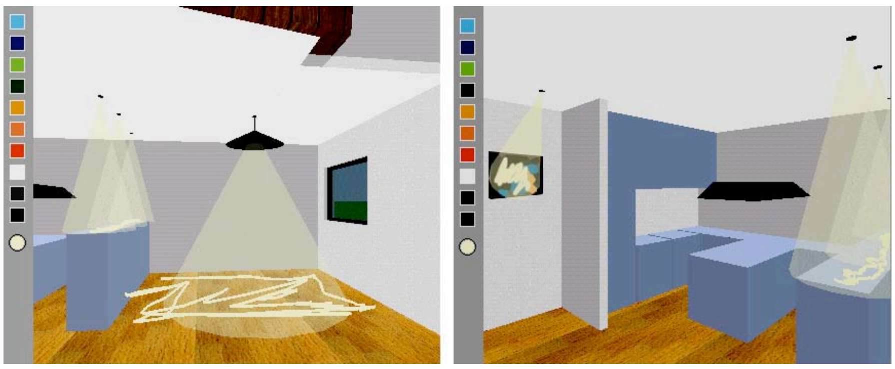


Figure 15. Light Pen system provides lighting fixture suggestions based on designer's sketches.

A third example is the Light Pen system (Jung 2003), which enhances the previously described Space Pen program with an expert advisor for architectural lighting (Figure 15). The designer uses a "light pen" tool to sketch desired lighting patterns on the interior surfaces of a 3-D design model. The program then analyzes the position, size, and intensity of these drawing marks and delivers this input to a rule based system for selecting lighting fixtures. The system proposes a set of lighting fixtures and locations that will produce the desired illumination. Light Pen shows how one might integrate a knowledge-based advisor in a 3-D sketching system.

The fourth and last example of linking with an external advisor is the LAN-Tools system (figure 16). LAN-Tools employs the Cocktail Napkin to recognize local area network (LAN) sketches, composed of symbols that represent basic LAN elements such as workstations, printers, and routers (Kuczun and Gross 1997). The designer's sketch is then given to a network design advisor that proposes modifications and improvements.

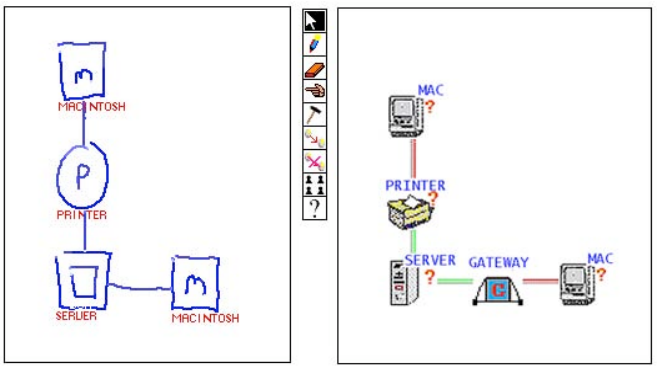


Figure 16 LAN-Tools – LAN design diagram (left) activates an advisor (right) that proposes design modifications, here inserting a gateway into the network.

2.3. RESOLVING DESIGNS – FROM ABSTRACT TO SPECIFIC

The previous two sections have examined the first two R's of drawing and design — Recording and Reasoning — activities of recording design information and applying domain knowledge and expertise to the design drawing. We turn now to the third R, Resolving — the process of specifying a design, from initial concept to detailed specification. Although we have stressed the importance of retaining a sketchy look and feel to accurately reflect the designer's level of commitment and decision-making in the early stages, sooner or later the designer must commit to decisions proceed to specify the design. Computational drawing systems must recognize and support this trajectory—allowing vague, ambiguous, and abstract representations at the outset, supporting the move toward more detailed and definite ones, and allowing for some interplay and movement in both directions. Resolving concerns the development of drawing over the design process from rough, sketchy, and abstract representations to specific, definite, and well-defined design proposals. Drawing can be used to beautify and specify design as in Electronic Cocktail Napkin and WebStyler, or translated into 3-D visualization with VR-Sketchpad (figure 17). The incremental formalization includes recognition of drawing symbols and configurations, to beautification of rectified figures, including the substitution of the diagram elements to specification drawing illustrated in the Electronic Cocktail Napkin.

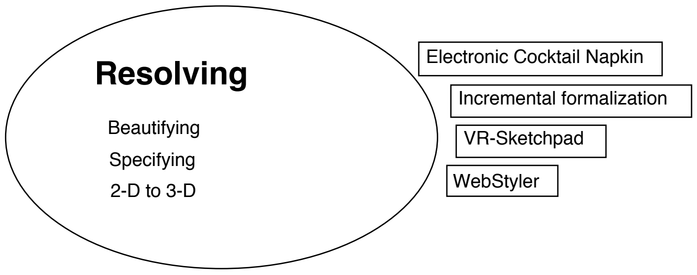


Figure 17. Resolving concerns drawing development & refinement during design.

2.3.1 Beautification

At the simplest level, the Cocktail Napkin's beautification scheme, in which each element type may have its own display method, moves the design along the path from crude sketch to precise drawing. When the program recognizes that the designer has drawn a stove, for example, in a kitchen, the beautified stove display method may replace the designer's sketchy stove with a dimensioned line drawing from a manufacturer's catalog (figure 18). However, this step may well involve additional selection criteria.

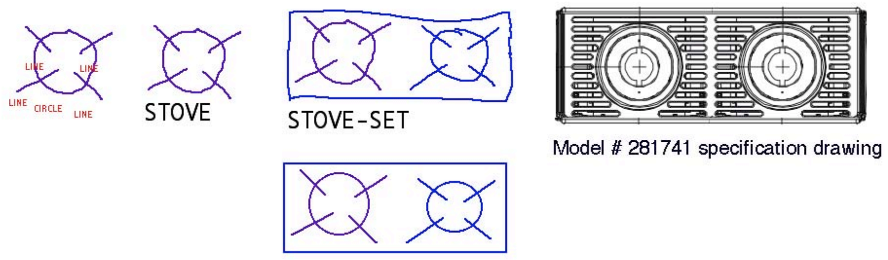


Figure 18. Incremental formalization, from diagram to CAD drawing replacement.

2.3.2 Toward specific commitments, 2-D to 3-D

The challenge of producing 3-D models from 2-D sketches has attracted a great deal of attention, but it is often seen as a purely a problem in computer graphics. In the context of design drawing we see moving from a 2-D sketch to a 3-D model, rather, as an example of specifying a design. A 2-D design sketch is an abstraction of a 3-D model.

Typically, the 2-D sketch does not in itself provide sufficient detail to produce a 3-D model, so information must be added in transforming the sketch to the model. For example our VR-Sketchpad program (Do 2002) (figure 19) transforms a floor plan sketch into a 3-D model. The program makes simple assumptions about the transformation: it extrudes walls and columns vertically, and it replaces floor plan symbols of furniture with 3-D models of furniture selected from the library.

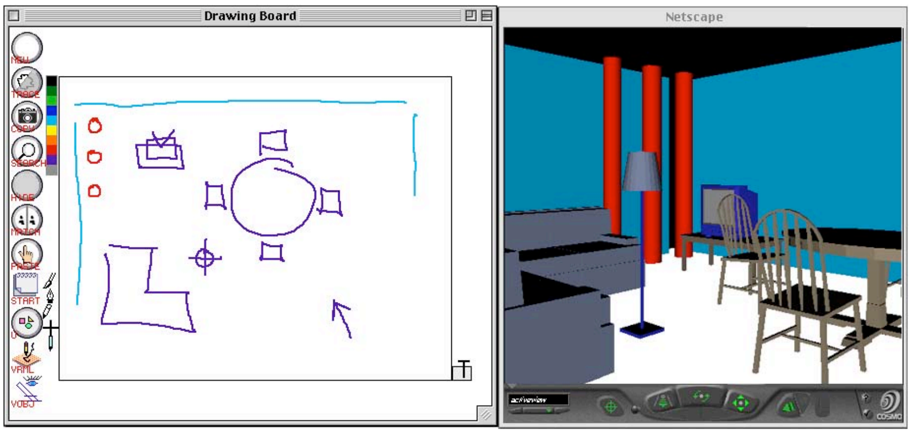
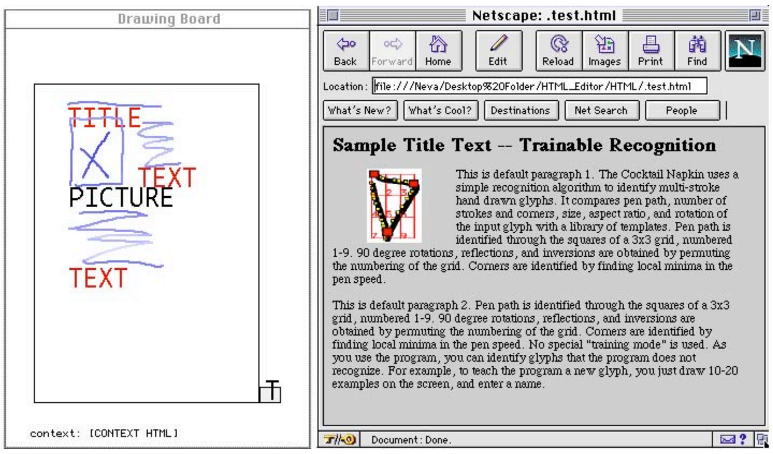


Figure 19. VR-Sketchpad generates specific 3-D models from rough 2-D sketches, extruding architectural elements and selecting furniture from a library.

Although the current VR-Sketchpad system simply maps floor plan symbols one-to-one with library elements, an intelligent system could mediate this selection process, depending on characteristics of the floor plan, previously selected furniture, and so on. We see the transformation from 2-D sketch to 3-D model not merely as a graphics challenge, but as an important case of specifying design details.

2.3.3 WebStyler — sketching Web page layouts

Web Styler (figure 20) also supports specification of a design from sketch to final product, although not in architecture, but Web pages (Gross 1996). A designer sketches a Web page layout by drawing elements to represent text headings, graphics, and other graphic design elements. The program then generates a dummy Web page based on this layout, and displays it in a browser. The designer can attach actual content to the layout by selecting text files and graphics and associating them with the elements of the Web page sketch. WebStyler then produces the page layout with actual content.



Sample Title Text -- Trainable Recognition

This is default paragraph 1. The Cocktail Napkin uses a simple recognition algorithm to identify multi-stroke hand drawn glyphs. It compares pen path, number of strokes and corners, size, aspect ratio, and rotation of the input glyph with a library of templates. Pen path is identified through the squares of a 3x3 grid, numbered 1-9. 90 degree rotations, reflections, and inversions are obtained by permuting the numbering of the grid. Corners are identified by finding local minima in the pen speed.

This is default paragraph 2. Pen path is identified through the squares of a 3x3 grid, numbered 1-9. 90 degree rotations, reflections, and inversions are obtained by permuting the numbering of the grid. Corners are identified by finding local minima in the pen speed. No special "training mode" is used. As you use the program, you can identify glyphs that the program does not recognize. For example, to teach the program a new glyph, you just draw 10-20 examples on the screen, and enter a name.

Figure 20. WebStyler generates Web pages from sketched layouts.

3. Discussion and Future Work

We have presented a raft of projects all centered on drawing in design. Individually, they explore topics from knowledge capture to visual analogy to simulating and evaluating performance. Taken together, the projects suggest various roles and uses for drawing, and ways of supporting these roles computationally. The specific computational support ranges from intelligent systems to management of informally expressed design rationale and annotations.

3.1 EXPERTISE IN MANAGING DESIGN PROCESS

The bulk of our efforts so far have dealt with the integration of design knowledge and the medium of drawing, that is, the bottom two ‘layers’ in the diagram of figure 1. The top layer, expertise in managing design process, arguably represents the most fundamental question in the theory and methods of designing: how do designers decide what to work on, when, in the course of doing a design project? Apart from what designers know about their particular domain, what ‘control’ expertise do they exercise in deciding how to devote their efforts? And what kinds of computational support might aid in applying this control expertise?

The “Right Tool at the Right Time” project looked at whether monitoring designers’ drawing acts could reveal useful information about their current purposes (Do 1998). Specifically, the project posited that by monitoring an architect’s drawing one might plausibly infer whether the designer was working on a spatial arrangement task, on resolving a lighting problem, or on calculating costs. Through a series of studies we found that designers (and students) could reliably infer task from drawing, and that specific drawing symbols and configurations were good predictors of task intentions. We then constructed the Right Tool Right Time

Manager. This program watches over the designer's sketches and—when it recognizes a symbol belonging to a specific subtask domain such as lighting or cost estimating, it proffers a supporting application (e.g., a lighting advisor or a cost calculator).

Although it only considers a small piece of the design expertise problem (what is the designer working on and what tools might be appropriate?) the Right Tool at the Right Time project does illustrate the possible relationship between design drawing and questions of expertise. Although one can—conceptually—distinguish the use of media, the application of knowledge, and the exercise expertise, in design practice the three are often intertwined.

3.2 OTHER MEDIA (GESTURE, COLLAGE, PHYSICAL MODELS)

We have taken a drawing-centered view of designing, focusing on the roles that drawing plays in design, the activities that designers do with drawings, and the ways in which computational tools might support those activities. However, we recognize that real-world designing is much richer. Designers employ other media as well in the course of designing, for example, physical models, collages, text, speech, drawing, and gesture interaction. In other, related, projects we have built systems to support the use of these media for designing, and we plan future work to integrate these efforts into our drawing centered model of design.

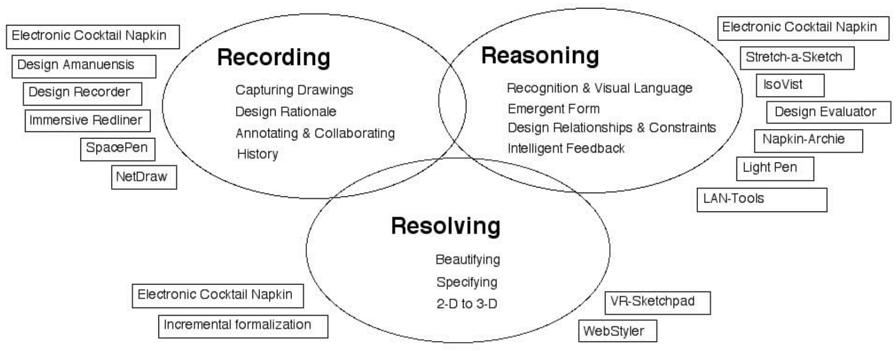


Figure 22. Software to support the Three R's of Design Drawing.

Acknowledgements

This research was supported in part by the National Science Foundation under Grants IIS-96-19856 and IIS-00-96138. 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

Contero González, M., F. Naya Sanchis, et al. (2003). CIGRO: A Minimal Instruction Set Calligraphic Interface for Sketch-Based Modelling. Computational Science and Its Applications - ICCSA 2003 . V. Kuma and P. L'Ecuyer. Springer.

Cross, N., H. Christiaans, et al., Eds. (1996). Analyzing Design Activity. New York, Wiley.

Davis, R. (2002). Sketch Understanding in design: Overview of Work at the MIT AI Lab. Sketch Understanding, AAAI Symposium. R. Davis, J. Landay and T. F. Stahovich., American Association for Artificial Intelligence**:** 24-31.

Do, E. Y.-L. (1998). The Right Tool at the Right Time: Investigation of Freehand Drawing as an Interface to Knowledge Based Design Tools, Georgia Institute of Technology.

Do, E. Y.-L. (2002). "Drawing Marks, Acts, and Reacts, toward a computational sketching interface for architectural design." AIEDAM, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, **16**(3): 149-171.

Do, E. Y.-L. and M. D. Gross (1997). Tools for visual and spatial analysis of CAD models. Computer Assisted Architectural Design Futures '97. R. Junge. Dordrecht, Kluwer Academic Publishers**:** 189-202.

Do, E. Y.-L., M. D. Gross, et al. (1999). Drawing and Design Intentions -- an Investigation of freehand drawing conventions in design. Design Thinking Research Symposium, Cambridge MA.

Falkenhainer, B., K. D. Forbus, et al. (1990). "The Structure Mapping Engine." Artificial Intelligence **41**(1): 1-63.

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

Gaver, W. W., J. Beaver, et al. (2003). Ambiguity as a Resource for Design. Conference on Human Factors (CHI 2003). Fort Lauderdale, FL, ACM**:** 233-240.

Goldschmidt, G. (1994). Visual Analogy in Design. Cybernetics and Systems '94,. R. Trappl. Singapore, World Scientific**:** 507-514.

Gross, M. D. (1994). Stretch-A-Sketch, a Dynamic Diagrammer. Proceedings of the IEEE Symposium on Visual Languages '94. A. Ambler, IEEE Press**:** 232-238.

Gross, M. D. (1996). "The Electronic Cocktail Napkin - working with diagrams." Design Studies **17**(1): 53-70.

Gross, M. D. (2001). Emergence in a Recognition Based Drawing Interface,. Visual and Spatial Reasoning II. B. T. J. Gero, T. Purcell. Sydney Australia, Key Centre for Design Cognition and Computing**:** 51-65.

Gross, M. D. and E. Y.-L. Do (1995). Shape Based Reminding in Creative Design. Global Design Studio: Computer Aided Architectural Design Futures '95. M. Tan and R. Teh. Singapore. **2:** 1-11.

Gross, M. D. and E. Y.-L. Do (1996). Ambiguous Intentions., ACM Symposium on User Interface Software and Technology (UIST '96). Seattle, WA:183-192.

Gross, M. D., E. Y.-L. Do, et al. (2001). The Design Amanuensis : an Instrument for Multimodal Design Capture. Proceedings Computer Aided Architectural Design Futures 2001. de Vries, . v. Leeuwen and Achten.., Kluwer **:** 1-13.

Gross, M. D., J. Lewin, E, Do, K. Kuczun, and A. Warmack (1996). Drawing as an Interface to Knowledge Based Design, Colorado Advanced Software Institute.

Gross, M. D., C. Zimring, et al. (1994). Using Diagrams to Access a Case Base of Architectural Designs. AI in Design '94. J. Gero. Lausanne, Kluwer**:** 129-144.

Igarashi, T. and J. F. Hughes (2001). A Suggestive Interface for 3D Drawing. Symposium on User Interface Software and Technology (UIST) ACM**:** 173-181.

Jung, T., E. Do, et al. (2002). Sketching Annotations in 3D on the Web. ACM Conference on Human Factors (SIGCHI), Minneapolis, ACM .

Kuczun, K. and M. D. Gross (1997). Local Area Network Tools and Tasks. ACM Conference on Designing Interactive Systems 1997. Amsterdam**:** 215-221.

Kurtoglu, T. and T. F. Stahovich (2002). Interpreting Schematic Sketches Using Physical Reasoning. AAAI Spring Symposium on Sketch Understanding. R. Davis, J. Landay and T. Stahovich. Menlo Park, CA, AAAI Press**:** 78-85.

Landay, J. a. M. B. (2001). "Sketching Interfaces: Toward More Human Interface Design." IEEE Computer: 56-64.

Leclercq, P. and R. Juchmes (2002). "The Absent Interface In Design Engineering." Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM) **16**(3): 219 - 227.

Mankoff, J., S. E. Hudson, et al. (2000). Providing Integrated Toolkit-Level Support for Ambiguity in Recognition-Based Interfaces. Proceedings of the Human Factors in Computing (SIGCHI) Conference. The Hague, ACM Press**:** 368-375.

Oh, Y., M. D. Gross, et al. (2004). Design Evaluator: critiquing freehand sketches. Generative Computer Aided Design Systems. Pittsburgh, Carnegie Mellon University**:** in review.

Oviatt, S. and P. Cohen (2000). "Multimodal Interfaces That Process What Comes Naturally." Communications of the ACM **43**(3): 45-53.

Qian, D. and M. D. Gross (1999). Collaborative Design with NetDraw. Computer Aided Architectural Design Futures '99. G. Augenbroe and C. Eastman , Kluwer.

Saund, E., D. Fleet, et al. (2003). Perceptually-Supported Image Editing of Text and Graphics. ACM conference on User Interface Software Technology. (UIST)

Saund, E. and T. P. Moran (1994). A Perceptually-Supported Sketch Editor. ACM Symposium on User Interface Software and Technology, (UIST) .

Schon, D. A., and Wiggins, G. (1992). "Kinds of Seeing and their functions in designing." *Design Studies* **13**(#2): 135-156.

Sutherland, I. (1963). Sketchpad - a Graphical Man-Machine Interface, M.I.T.

Suwa, M. and B. Tversky (1997). "What architects and students perceive in their sketches: A protocol analysis." Design Studies(18): 385-403.

Tversky, B. (2002). What do Sketches say about Thinking? AAAI Spring Symposium Series -- Sketch Understanding. T. Stahovich, J. Landay and R. Davis.

Ullman, D., S. Wood, et al. (1990). "The Importance of Drawing in the Mechanical Design Process." Computers and Graphics **14**(2): 263-274.