Using diagrams to access a case library of architectural designs

M. GROSS
College of Architecture and Planning
University of Colorado
Boulder, CO 80309-0314 USA

C. ZIMRING, E. DO
College of Architecture
Georgia Institute of Technology
Atlanta, GA 30332 USA

Abstract
We describe the linking of a ‘computer as cocktail napkin’ program that interprets hand-drawn sketches and diagrams with Archie III, a case based design aid, to support case-based reminding in conceptual design. Each case in Archie’s library includes stories, problems, and responses indexed and accessed by carefully chosen features. In addition to text, photographs, and drawings, many items in Archie’s case library are illustrated by simple diagrams. We have added these diagrams to Archie’s indexing scheme, so a hand-drawn sketch can be used to retrieve items tagged with similar diagrams.

1. Introduction - why link sketching with a case based design aid?

In many disciplines, but especially in architecture, designers use sketches and diagrams during conceptual design to explore and communicate alternatives in a rough and rapid fashion. In later design stages, a few promising sketches are developed into more carefully made drawings with precise dimensions. Sketching is quick; it forces the designer to focus on a small number of elements and relationships; and it does not involve great effort or precision.

Despite the benefits of sketching for conceptual design, today’s CAD tools support diagram and sketch-making only poorly. Most CAD programs require a commitment to precise dimensions and placements that designers rightly avoid during conceptual design. Even programs that do offer sketch-making capabilities (such as Alias’s “Sketch”) do not attempt to recognize or interpret the designer’s hand drawn input, so they do not engage the...
machine’s computational power. The failure of CAD tools to support making, management, and machine recognition of sketches and diagrams motivates designers to stick to paper for conceptual design.

If CAD tools fail to support conceptual design, so do most programs for simulation and evaluation. Although it is possible to build qualitative evaluation programs that support conceptual design decision making, today most evaluative tools such as simulation programs serve the later phases of design development and optimization, and they require users to input quite detailed design descriptions. In contrast, a case based advisor is well suited to supporting conceptual designing. A case library can provide stories about previous design successes and failures, helping a designer identify features to include or problems to avoid in a new design. A case-based design advisor can provide information relevant to the task at hand, including text, drawings, photos, movies and sound, if the designer can use the case library’s indexing scheme to identify salient features of the design task.

**Tools must be integrated in design environments.**
A common difficulty with applying case based reasoning as well as other AI approaches in design has been the separateness of the design tool from the designer’s working environment. For example, previous versions of the Archie case based design aid have been largely reference tools, entirely external to the designer’s actual working environment. The designer must first realize that a certain piece of information is needed, negotiate the case based design assistant to find that information, and carry the information back to the design environment where it is to be employed. Each step of this process -- realizing that knowledge is needed, finding the knowledge, and applying the knowledge in the design -- imposes a barrier to more effective designing.

Integration of argumentation, design information, and access to stored cases in the designer’s working environment (a structured “construction kit” editor) is prominent in the work of Fischer and his colleagues. In their Janus system, for example, knowledge based critics detect particular conditions in the design and automatically trigger advice linked to an issue-base of design argumentation (Fischer and Morch 1989). Likewise in Catalog Explorer, case retrieval is linked with functional design specifications and with the construction kit where designs are assembled (Fischer and Nakakoji 1991). We agree that integrating advice into the design environment is essential in order for a designer to be able to make effective use of stored information. For a case based design assistant this
integration must include (1) retrieving items in the library based on the designer’s actions (2) the ability to copy partial designs from the case library and adapt them in the design, and (3) retaining links from copied design fragments to the case library in order to provide continuing reminders of relevant stories.

Although JANUS and its successors integrate knowledge based critiquing and retrieval of examples in a design environment, in these programs the design environment employs a structured editor (“construction kit”) in which users select items from a palette and assemble them in a work-area on the screen. This method of designing, common to most CAD editors, is more appropriate for design development and routine design, where the universe of components is already determined, than for conceptual exploration. Although the sketches and diagrams made in conceptual design may eventually comprise the same elements that appear in more structured representations, the unstructured and imprecise nature of sketch-making appears preferable in early design.

In summary, a key reason to support recognition and interpretation of hand drawn sketches is to link AI-based evaluative tools such as critics and advisors with conceptual design thinking. A designer who has made the effort and commitments to model a design precisely in a CAD environment will be understandably reluctant to make major changes suggested in a subsequent evaluation. If a designer could access case based advice, critiquing, and other evaluative assistance in the early conceptual phases of design this capability of earlier evaluation and modification could have a greater impact on the design.

To explore this idea we have developed a working prototype (in Macintosh Common Lisp) that links machine recognition of simple diagrams with items in a case library of designs. Our prototype has two distinct components: an environment for making and managing design diagrams, and a case based design assistant. The sketching part of our program includes a trainable recognizer for hand drawn diagrams and a mechanism for graphical search. The case based design assistant is Archie, which accesses a catalog of library and courthouse designs as well as post-occupancy evaluation (POE) data about the designs. In the following two sections (2 and 3) we describe these two main components of our prototype. In section 4 we explain our scheme for diagramming problems, responses, and stories in Archie’s case library, how a designer’s hand drawn sketching retrieves items from the case library. We conclude by identifying tasks for the next round of prototype, which will more closely integrate Archie in a design environment.
2. Recognizing and interpreting diagrams

The sketching part of our program is an environment for making, managing, and recognizing hand drawn diagrams (Gross 1993). Figure 1 shows a screen snapshot of this “electronic cocktail napkin” program. The program recognizes multi-stroke glyphs, matching shape, stroke count, and number of corners, matching input against templates stored in a catalog. An initial built-in set of glyphs includes numbers and capital letters, geometric shapes, and dotted, dashed, and solid lines. The designer can easily and interactively train the program to recognize new shapes, extending the template library. In addition, the program uses a small list of predicates (such as above, contains, near) to identify spatial relations between the elements of a diagram. The designer can program the system to recognize characteristic configurations of glyphs, such as a tree-diagram or a bubble-diagram. Then the program can search a database for occurrences of a certain diagram or diagram fragment. Thus, graphical search with hand-drawn diagrams as input is used to find items in Archie’s case library.

![Figure 1. Screen snapshot of the “electronic cocktail napkin” program.](image)

Recognizing and interpreting diagrams is not a new topic, but one that has languished until recently for over a decade. In the early 1970’s Negroponte and others at the MIT Architecture Machine Group worked on machine aids to sketch recognition (Negroponte 1973). Though promising, this line of research was largely abandoned as most computer interfaces during the 1980’s came to employ mouse and windows interfaces. Recent development of “pen based” computing hardware has seen a re-emergence of this area of research in human-computer interaction, including techniques for recognizing hand-drawn input (Goldberg and Richardson 1993; Zhao 1993) and the use of shared drawing surfaces.
A trainable recognizer for hand drawn sketches

The “cocktail napkin” sketch program reads raw coordinate and pressure data from a WACOM, Inc. digitizing tablet at 1200 baud, producing a “raw glyph” data structure that contains a point list, stroke count, and bounding box. Points of the raw glyph are reduced to a 3x3 grid overlaid on the glyph’s bounding box, and the glyph is more simply described as a sequence of the squares (numbered 1-9) through which the pen moved. In addition, when the pen slows down to start or end a glyph or to round a corner, the points sampled are closer together and can be identified as corners. Figure 2 shows the raw coordinate points, the 3x3 grid inscribed in the bounding box, and the corners identified for several simple glyphs.

![Figure 2. Features of simple glyphs.](image)

To identify a hand-drawn glyph, the program compares the input with a stored library of templates. To allow for variation in drawing, each template stores a number of allowable paths through the grid and other parameters. The library of templates consists of
previously trained glyphs and is extended automatically as the designer provides new examples. The program first looks for an exact match between the input glyph and the templates, but if none is found it gradually relaxes the match criteria. If more than one template matches the input, the program carries the ambiguity, which can be resolved later with the help of additional context. The user can interactively correct the recognition process when it fails, and the template library is updated to include the correction. Figure 3 summarizes the procedure used to match hand drawn input with glyph templates in the library.

Figure 3. Summary of the low-level glyph-matching procedure.

**Spatial relations**

A small list of binary spatial relations is programmed into the recognizer in the form of predicates. These enable the program to determine, for example, whether two glyphs are near or far, concentric, overlapping, or disjoint, and roughly the same size or different. By applying these predicates over pairs of glyphs, the program can produce a description of spatial relations present in a particular configuration. Only relations between pairs of glyphs are considered; conjunctions and disjunctions can be used to construct more complicated relational descriptions. Because the simple diagrams we are interested in have small numbers ($N < 10$) of glyphs, combinatoric explosion has not been a serious problem. In addition, certain relations are associated with certain classes of glyphs; for example, lines ‘connect’ and ‘intersect’ but do not ‘overlap’. Figure 4 shows the program’s analysis of spatial relations in a simple diagram; the program has identified the
sequences of letters as words, and the bubbles containing words as diagram elements named ‘court’, ‘buffer’, and ‘wait’.

![Diagram](image)

Figure 4. Analysis of spatial relations in a simple diagram.

Diagrams as configurations of glyphs and relations

Representing a diagram as a set of glyphs and spatial relations suffices to describe configurations such as words, tree diagrams, and bubble diagrams. A set of higher-level recognizers identify configurations, finding sets of glyphs that satisfy certain spatial relationships. For example, a ‘word’ is a collection of letters more or less the same size, arranged more or less horizontally and next to one another. A ‘room’ (in a floorplan bubble diagram) is a ‘word’ contained in a box or a circle. In general, a higher-level recognizer identifies collections of glyphs $g_1, g_2, \ldots, g_j$ such that relations $R_{i,j,k}(g_i, g_j)$ hold among pairs of glyphs. The relations $R$ include spatial predicates as well as relations that specify the glyph types. For example, a simple description of a tree diagram includes the following relations:

![Tree Diagram](image)

Figure 5 - “tree diagram” relations.

Which is to say, a ‘tree’ is a collection of circles and lines such that each line connects two circles, one above the other.

The higher-level recognizers that find configurations of glyphs can also resolve lower-level ambiguities. For example, lacking context, the lower-level glyph recognizer cannot decide
whether a circle is meant to be a circle, the letter “o” or the number zero. However, when a higher-level recognizer finds the circle next to two letters, then it declares the glyph to be the letter “o”. On the other hand, if the circle is found in a tree diagram, it remains a circle.

The higher level tree recognizer can be used within the sketch program alone to retrieve pages that contain tree-diagrams from a catalog of previously made diagrams. This same graphical search and retrieval provides the link to Archie’s case library. The program can search the Archie case base for various kinds of diagrams. These include (1) floorplan bubble diagrams containing key adjacencies, overlaps, and containments; (2) diagrams indicating lighting, visual access, and acoustics; (3) conceptual hierarchy diagrams indicating building organization; and (4) detailed arrangement of objects, such as the configuration of furniture in a courtroom.

In addition to a trainable recognizer and graphical search facility for configurations of glyphs, the “cocktail napkin” sketching environment provides other features that we think will prove useful for conceptual design. These include a simulation of tracing paper, gestural versions of the most frequently used commands, a catalog for storing sketches, and a way for two or more users to share the drawing surface.

A note on generality of diagrams
The general applicability of our approach hinges on two observations. First, diagrams are made up of a relatively small vocabulary of symbols and spatial relations. If more or less the same symbols are used across many disciplines, it will be possible to work with single library of low level glyphs, composed in different ways. On the other hand, if diagrams are made of a large vocabulary of symbols, then it will be more difficult to distinguish diagrams that are subtly different. To lend weight to our hunch that the vocabulary is small, we photographed approximately 50 diagrams on blackboards around our campus, in the departments of electrical engineering, mechanical engineering, architecture, and biochemistry. We found that, to be sure, some disciplines use special symbols such as “transistor.” But for the most part the diagrams contain the same basic circles, arrows, lines, wiggles, and alphanumeric labels combined in fairly simple and similar ways.

The second observation is that diagramming is not highly idiosyncratic; people make more or less the same diagrams. To test this observation we asked 60 undergraduate architecture students to make sketches of ten slides, giving them 15 seconds to look at each image and 30 seconds to make a diagram. Five slides showed plan or elevation drawings, the other
five showed photographs of buildings or building details. For each slide the sketches were strikingly similar, containing largely the same elements and spatial relations. Although for a few images there were two or three alternative ways that students made the sketches, the differences were for the most part surprisingly small. This experiment suggests that designers will be quite consistent in their use of diagrams.

3. Archie - a case based design aid for architecture

Archie III is a case based design assistant (CBDA) for architectural design* (Domeshek and Kolodner 1991, 1992). Like other CBDA programs, Archie contains a case base of designs including both good and bad exemplars annotated with stories that describe key design features and how they function in the building. Archie’s initial case library contains ten cases of courthouses and libraries. Each case comprises over twenty-five stories covering a range of different design concerns. Each case is constructed from drawings, photographs and text obtained from the architect, from visits to the building, and most importantly, from post-occupancy evaluations (POE’s).

Post occupancy evaluation is a conventional method used to evaluate the performance of a building design. POE proceeds by structured interviews with building users, and it frequently reveals ways a building is actually used that were not considered by the designer. Post occupancy evaluations are increasingly included in the routine building delivery process of major governmental and corporate entities, such as the California Department of Corrections, Health and Welfare Canada, and the US Postal Service.

Unfortunately, POE’s are usually not carried out by the architect, but by a separate consulting firm. POE data is collected and presented to the client, and perhaps an immediate problem is solved; however the lessons learned from evaluation often do not reach the designer for consideration in future work. Therefore, one goal of the Archie project is to render knowledge gained in post occupancy evaluations more accessible to designers who are in the early stages of work on similar design problems. We also believe that a case based design assistant like Archie can provide a mechanism for corporate

* We refer to it simply as Archie.
memory in a design firm; it can help also coordinate various stakeholder interests in a complex building design.

A major part of the work in Archie has been to structure post-occupancy evaluation and other data in a format amenable for inclusion in Archie’s case base. Both text and graphic data must be entered into the case-base. Each problem, response, and story must be indexed using a set of features developed specially for the Archie case library, and related items must also be cross-linked for browsing. Text items in each case are classified as “problems,” “responses,” or “stories.” Each item must be clearly and concisely written, labeled with an explanatory title, and summarized in a one-paragraph synopsis. Graphic material (plans, photographs, and drawings) associated with each text item is scanned and manipulated so that it will appear clear on the screen.

Figure 6. Stories in Archie are linked to a building floorplan—circles show their location.

The two main ways to find information in Archie’s case library are (1) access through an index of the features of stories, and (2) browsing from story to story, or from a graphic image (e.g. a floorplan) as shown in figure 6. Initial access to items in Archie is through an index of key features to be found in the cases. The index is organized along five major dimensions: systems, components, design issues, stakeholders, and life cycle concerns. These dimensions, employed also in other case based design assistants, have specific interpretations in Archie. ‘Systems’ include the circulation, HVAC, and structural systems; ‘components’ identify specific elements such as rooms, doors, elevators; design issues
include noise, privacy, access for physically challenged users; “stakeholders” include the client, building management, and various classes of users; and “life cycle concerns” include maintenance and repair. A particular story is likely to be indexed by several categories of features. For example, in one case a story explains how the high internal pressure of the HVAC system results in the rear private staff doors hanging open, causing serious problems for the circulation system, which is based on the idea that staff occupy a separated, isolated zone.

In the standard interface to Archie, the designer identifies retrieval keys by choosing them from a set of menus. For example, the designer can identify issues such as “access to courtrooms,” or “security & safety,” systems in the design such as “circulation” or “building structure”, or provide descriptors to limit search, such as “urban setting,” or “area greater than 200,00 square feet.” Archie’s retrieval machinery then provides items in the case base that have been tagged as potentially relevant to these concerns. In addition to the index that identifies major features of each story, links between items in the case base provide a way for a designer to browse from case to case without revisiting the index each time.

4. Indexing Archie’s case library with diagrams

Most of the stories in Archie’s case base include some graphic information -- a plan fragment or a small diagram that illustrates or explains the principle described in the story. These illustrations are made using a draw program or they are scanned in from printed documentation about the building. To access stories from hand drawn sketches, we augment them with a simplified diagram of the illustration, composed of a small number of basic glyphs. (We will use ‘illustration’ to refer to the more detailed picture and ‘diagram’ to refer to the crude, simplified representation). These diagrams bear the same relation to the illustration that the title and keywords bear to the story text -- they summarize and highlight an important point or relationship, but they do not tell the whole story. Many of these diagrams are simple ‘bubble diagrams’ made of crude ovals, linked by short line segments or arrows, and often labeled. In bubble diagrams, certain relationships are most important -- adjacency, containment, overlap, and connections with lines, whereas size and shape are often incidental.
For example, figure 7 illustrates the story “noise disturbance due to poor design of doors.” In this story, two sets of doors were placed to act as a sound buffer between the public area and the court room; however, because the doors have no windows, people constantly go in and out to check on courtroom proceedings; the door hardware bangs loudly each time these doors are opened, disrupting court proceedings. Figure 7 shows both the formal illustration (at left) for the story as well as the simple diagram used for indexing from sketch input.

![Diagram of a court room with buffer areas and a sound disturbance]

Figure 7. An illustration from a San Jose Courthouse story with corresponding diagram.

As figure 7 shows, the diagram keys are considerably simpler than the figures used to illustrate the story. The illustration (at left) aims to provide both a visual summary of the problem described in the story and to portray a specific layout from the case; the illustration often adds information not found in the text. On the other hand, the diagram used as a key for sketch-based indexing shows only one aspect of the story -- in this instance, the concept of a buffer zone between two communicating areas. The diagrammatic elements are simple: a double-headed arrow and three bubbles, with the label “buffer.”

Figure 8 illustrates the story, “benches in the waiting room are well designed for visitors.” The story explains that the high backed oak benches in the Bristol County Courthouse waiting room are comfortable, and provide some acoustic privacy for visitors’ conversation. The diagram on the right is a simplified version of the illustration. This diagram includes a special glyph for sound (three or four concentric arcs) and the glyph for “seated person.”

![Diagram of high back benches with sound and seated person glyphs]

Figure 8 - illustration and diagram: “benches in waiting area provide acoustic privacy”
The illustration and diagram in figure 9 are associated with a story about the location of the building entrance with respect to the street and the parking area: “the main entry faces the parking lot in order to meet the needs of staff and the public arriving by car. The back of the courthouse faces the main street and town center; visitors approaching the building on foot must walk around the building, passing its service prisoner entrance...” The elements of the diagram are two parallel lines, labelled “street,” bubbles labelled “parking” and “building,” and arrows that indicate the building entrance and pedestrian access. Higher level recognizers identify the two parallel lines as a street or path, and the arrow pointing into the building as potentially an entrance.

Retrieval from sketching

The premises for this first prototype are that during early design, sketching is a natural medium for exploration, and case-based reminding of important issues and relationships even at the early stage can save time and trouble later. Stories from Archie’s case base are retrieved in response to simple hand drawn diagrams. As the designer sketches, the program identifies key elements and relationships, matches them with diagrams in the case base, and brings relevant stories to the designer’s attention. Of course, one could argue that the sketches are not needed; merely a specification of the elements and relations they embody. However, we believe there is an intrinsic value added by allowing the designer to express these relations through the natural medium of sketching.

When “reminding” is turned on, the sketch program searches the case base for diagrams that match parts of the designer’s evolving sketch. The match can be made more or less exact. For example, suppose a designer is thinking about acoustics and draws the glyph for “sound” (see figure 8). If inexact matches are permitted then the presence of a sound
glyph in the designer’s sketch recalls all diagrams containing the sound glyph and retrieves their associated stories. On the other hand, in a more exact match, only diagrams that contain both a seated person and a sound glyph will be retrieved. Likewise, a bubble with an arrow pointing into it is often used to indicate “entrance.” If inexact matching is permitted, then all diagrams containing this configuration will be retrieved along with their stories; on the other hand, if the label “courtroom” is included, then the sketch will only retrieve stories about courtroom entrances.

Our integration of diagram making and the Archie program is quite crude, as might be expected when linking two already-built, stand-alone programs. Each story that is diagrammatically indexed points to one or more diagrams in a separate catalog. The diagram(s) can be displayed whenever the story is retrieved, either by a keyword search or by navigation from elsewhere in the case base. The sketch program’s graphical search routine is set up to search the catalog of diagrams. When it finds a matching diagram, the sketch program signals Archie to retrieve the associated story. Thus the sketch program and Archie remain distinct modules sharing a data structure and they communicate only in the simplest possible way.

5. Discussion and further work

Our prototype is simple, but it works. We believe it shows an appropriate way to integrate powerful AI tools such as case based reminding with environments for conceptual design. In linking our case based design aid with a sketching program, we feel we have overcome one of the features of Archie that architects find most limiting: its text and menu-based interface for initiating retrieval. Although we have illustrated this approach with a case base about building design, we are certain that we could tell a similar story about other design domains.

Linking Archie with a sketch program has clarified for us the importance of integrating the AI tools and techniques in environments that support and enhance regular design activities such as making drawings, diagrams, and sketches. If AI tools are to be effective assistants to designers, then they must be able to communicate with designers with a minimum of special apparatus. Embedding the tool in an environment for making designs offers the best hope that it may be used.
Several areas of further work seem ripe for investigation. One area involves our plan to extend the diagram indexing scheme to include more of the case base. This may require the development of an understanding of users’ mental models of Archie’s cases, which would lead to more formal conventions for making the diagrams. In addition, some usability testing would help us assess the value of retrieval-by-diagram. Another area of research would focus on strengthening the diagram recognition program. This includes training the program with a somewhat larger vocabulary of glyphs, and adding spatial relationships. In addition, the list of spatial relations could be expanded. We have not built an interface for programming the “higher-level recognizers” by example, and this might be a useful feature. A third area of work involves developing the sketch program into more of a design environment, and integrating it more fully with the Archie advisor. For example, the designer should be able to find relevant stories in the case base, copy their diagrams into the sketching program, enhance, embellish, and develop the diagrams into more rigorous drawings, while, watching in the background, Archie displays examples of relevant designs. As part of a smoother transition between the stages of conceptual diagram and design development, we would like to program the sketching environment to maintain the spatial relations in a diagram -- such as adjacency, proximity, relative size, and alignment constraints -- as the user moves from buble diagrams to more precise design representations.

Acknowledgements
The Archie project, and the related MIDAS aircraft design project, have involved many people, and benefited from the contributions of many others not directly involved. Janet Kolodner and Eric Domeshek have served as co-principal investigators along with Craig Zimring on the Archie Project, and Domeshek has done much of the coding of the CBR shell. Michel Conan is co-investigator on the EduTech project aimed at developing Archie as a educational tool. In addition to Ashok Goel and Richard Billington who helped shape these projects at the earliest stages, we have been fortunate to work with CS students Anna Zacherl and Vijaya Narayanan, architecture students Ellen Do, Husam Khalil, Ali Malkawi, Ameen Farooq, and Osman Ataman, and aircraft designers Marcia Herndon and Andrew Bennett. This work has been supported in part by EduTech Project contract N00014-91-J-4092, and a grant from the National Science Foundation DDM-9313186. Some of the tool kit work was supported by Lockheed Aeronautical Systems Company.

References


