GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

NOTICE OF PROJECT CLOSEOUT

Closeout Notice Date 06/12/96

Project No. C-36-681
Project Director KOLODNER J L
Center No. 10/24-6-R7365-0A0
School/Lab COMPUTING

Sponsor NORTHWESTERN UNIVERSITY/EVANSTON, IL

Contract/Grant No. 0650 355 A66 GT
Contract Entity GTRC

Prime Contract No. N00014-91-J-6092
Title REASONING WITH VERY LARGE DATABASES

Effective Completion Date 950614 (Performance) 950814 (Reports)

<table>
<thead>
<tr>
<th>Closeout Actions Required</th>
<th>Y/N</th>
<th>Date Submitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final Invoice or Copy of Final Invoice</td>
<td>Y</td>
<td>950508</td>
</tr>
<tr>
<td>Final Report of Inventions and/or Subcontracts</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Government Property Inventory &amp; Related Certificate</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Classified Material Certificate</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Release and Assignment</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

Comments

Subproject Under Main Project No.

Continues Project No.

Distribution Required:

<table>
<thead>
<tr>
<th></th>
<th>Y/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Director</td>
<td>Y</td>
</tr>
<tr>
<td>Administrative Network Rep</td>
<td>Y</td>
</tr>
<tr>
<td>GTRI Accounting/Grants and</td>
<td>Y</td>
</tr>
<tr>
<td>Contracts</td>
<td></td>
</tr>
<tr>
<td>Procurement/Supply Services</td>
<td>Y</td>
</tr>
<tr>
<td>Research Property Management</td>
<td>Y</td>
</tr>
<tr>
<td>Research Security Services</td>
<td>N</td>
</tr>
<tr>
<td>Reports Coordinator (OCA)</td>
<td>Y</td>
</tr>
<tr>
<td>GTRC</td>
<td>Y</td>
</tr>
<tr>
<td>Project File</td>
<td>Y</td>
</tr>
<tr>
<td>Other</td>
<td>N</td>
</tr>
</tbody>
</table>

NOTE: Final Patent Questionnaire sent to DODT.
May 14, 1996

Patrick Brockhagen
Northwestern University
Institute for the Learning Sciences
1890 Maple Avenue
Evanston, IL 60201

Dear Mr. Brockhagen,

This letter plus the enclosed papers comprise the final report for Georgia Tech's part of the ARPA project, N00014-91-3-4092 (Reasoning with Very Large Case Bases), ONR Project No. N00014-92-J-1234 (A Case Based Approach to Creative Design).

Previous reports have reported on the first three years of work on the project. During the last year of the project (June, 1994 through June, 1995), we expanded use of the case-based design aiding tool we developed to additional domains (sustainable technology and development, industrial design, skyscrapers, handicapped access) and to additional types of users (learners). Several conference papers report on these efforts. More detail can be found below.

Case-based design aids were designed for a user population of expert practitioners. Learners, however, do not yet know the vocabulary for a new design domain, nor are the kinds of search and interface capabilities developed for practitioners easy for them to use. Thus, our emphasis in making Case-Based Design Aids (CBDA's) into Case-Based Learning Aids (CBLA's) has been to focus on changing three aspects of the system design: content, interface and search functionalities, and integration.

Content: The original CBDA's contained actual design cases described in technical terms that experts could easily understand. The cases were indexed using such terms as well. Since students are novices in the process of becoming experts, we have modified the content in three ways to make the case libraries better support learning. One is by incorporating cases of a tutorial nature into the library. A tutorial case looks just like a real case, except that the materials refer to the design of a generic or ideal artifact of the domain, and are introductory in nature. A prototype CBLA with these attributes, to be used for a tall building design studio at Georgia Tech, has been built. It contains a detailed description of how to design various systems (e.g., electrical, HVAC, mechanical, conveyance) of a generic tall building in the form of a tutorial example, as well as real cases.
The second change in content is to make the indexing structure of the case library reflect actual practice in the domain of instruction, both by using standard terms in the indexing vocabulary and by structuring the vocabulary to reflect how knowledge is typically structured in the domain. For example, the indexing vocabulary used in the aforementioned case library reflects the terms and organization found in the most popular handbook used by the construction industry.

The third way in which content is being changed is by changing the structure of the knowledge in the case library. For example, in the CBLAs that we are constructing for use by middle school students, case information is organized in terms of multi-media overviews that are linked to specific and general elaborations of significant aspects of a case. This is a simplification as well as a generalization of the original knowledge structuring in CBDA.s.

Interface and search functionalities: Interface and search functionalities have been simplified. One reason for this is what we found from experience: users tended to use only a small subset of the available features most of the time. Besides, we want students to focus more on the content than the "bells and whistles" of a fancy interface. One important way in which this goal is being realized in the CBLAs being constructed for middle school students is by making text in the information presentation windows "hot" and linking important pieces of text to detailed elaborations. This way, a student gets to an elaboration of an idea from within the context of the use/application of that idea in the case being looked at. This model replaced the practice of using separate buttons to traverse semantic links in CBDA.s. Another important change is the provision of an "explain" functionality with the search facility. Thus, a student can ask for in-context explanation of an unfamiliar index term before he/she decides to use it in a search query. Finally, the search query that students will construct is in the form of a set of questions (e.g., Which devices perform the function of dialysis?) rather than as a logical expression containing of a set of index terms.

Integration into practice: CBDA's were to be integrated into the natural design practice of practitioners. For learners, however, practice means different things than for practitioners. Learners are often learning in the context of a course that has them engaging in a variety of activities. Thus, integration of a CBDA into the practices of a learner means integrating it into the curriculum. CBDA's were designed to be stand-alone systems that an expert will consult for solutions in the midst of a design session. CBLAs, designed for learners, are information resources that a student uses, along with a number of other tools and classroom activities, in the context of learning by solving problems. On occasions when these systems were made available as optional and separate (from other learning activities) resources, their use was minimal. Thus a need to specifically integrate their use into the curriculum.
We discovered this problem in the fourth year of the contract, and we are currently addressing it in several ways in further research funded by other grants and contracts. One way we address the issue is by integrating CBLAs with other tools. In one project aimed at middle schoolers, we are integrating CBLAs into an electronic workspace, called McBAGEL, that the students use for recording their problem solving activities. In another project, joint with Civil Engineering and Architecture, CBLAs are being integrated into a design problem solving assistant that will guide students along different possible paths one can take in tackling design-and-construction problems. Another strategy for addressing this problem is to integrate CBLAs with the pedagogy for a course. In our middle school learning-by-design project, CBLAs will be provided to students in the context of a specific pedagogical model of learning and instruction. This model, called problem-based learning, encourages students to engage in self-directed learning guided by their knowledge deficiencies that they themselves uncover during the course of collaboratively solving problems. Thus, this model sets up both a context and motive for effectively utilizing CBLAs for remedying their knowledge gaps. Within this context as well, we are preparing for students to use CBLA's to record their own design experiences for other students to benefit from. Case-based reasoning's cognitive model suggests that the activity of creating cases from their experiences will help students structure and index their experiences appropriately. The activity of using other students' cases should help students discover indexing terms that are relevant for the domain they are learning.

At present, we have four ongoing CBLA projects. These are the tall building (skyscraper) library for architectural students, a set of case libraries (of artifacts and experiments) to be used in a problem-based middle school science curriculum, a library of industrial design cases that is being incrementally built by students in an industrial design case analysis course offered every quarter, and a case library of design and construction cases to be used by civil engineering students.

The enclosed papers summarize the rest of the work accomplished on this project.

Sincerely,

Janet L. Kolodner, Principle Investigator
Professor, Computing and Cognitive Science
THE DESIGNERS' MUSE

Providing Experience to Aid Conceptual Design of Complex Artifacts

ERIC A. DOMESHEK
The Institute for the Learning Sciences, Northwestern University

AND

JANET L. KOLODNER
College of Computing, Georgia Institute of Technology
1. Conceptual Design

Over beer one night while visiting a high school friend who had grown up to become a NASA engineer, one of the authors fell into conversation about this friend's work on the design of a new telescopic camera for a space probe. It happened that the engineer was just beginning to think about how he would design the lens cover for his camera. As he sketched the important issues and outlined his initial design ideas, what unfolded was a glimpse of conceptual design. His problem gives a good idea of what typically goes on during that early stage of problem exploration, idea generation, and tradeoff analysis.

What were the important issues for his design problem? The camera needed a lens cover because it would contain delicate optics that must be shielded from dirt and contamination during early stages of the mission. During launch the cap would have to remain secure despite extreme vibration. In flight the cap and its mechanism would have to survive tremendous temperature extremes and gradients. Ultimately, despite the beating taken along the way, the mechanism would have to remove the cap from the camera as the probe approached its target. The mechanism would only have to work once, but it absolutely had to work, or NASA would have a billion dollar piece of scrap floating beyond the reach of any repairs.

There are two things to note about these deliberations. First, during this stage of design, he was discovering what was important about the problem. The process that leads to these discoveries, which we call problem exploration, is a process of playing scenarios through in the head to discover what was important about them. By knowing how things worked on other space probes, this designer was able to focus on a set of important issues. Second, the issues he was concerned with were considered at a fairly high and non-specific level. At this point in the design process, coping with dirt and vibration, accounting for temperature, and ensuring reliability were not hard quantified requirements.

What were some of his initial design ideas? He was pretty sure that he would build the lens cap out of a transparent material; then if the cap removal mechanism failed, the probe could still get some use out
of the camera. Having anticipated a possible failure (the cap staying jammed shut), he planned for a fallback option. He also expected that the cap would be spring-loaded so that the only operation required to remove the cap would be to unsecure it; once unsecured, the cap would simply pop open, and there would never be any need for a mechanism to close it up again. Having identified a constraint in the situation (one-time operation of the opening mechanism), he was able to exploit it to simplify his design.

Again, in order to understand conceptual design, we should note the kind of ideas he was generating. He was proposing strategies for coping with potential problems and exploiting problem constraints. He was not interested in the details of the design and its parts. He did not know or care about the transmissivity of any particular lens cap material, and he was not ready to start worrying about the spring parameters. What this means is that for this work, it did not matter very much that he had the skill to calculate torques or the knowledge to choose parts from catalogs. His primary leverage during this conceptual design process was familiarity with mechanical configurations in general and past space probes in particular.

This held true for what became the focal problem in his initial design: the tradeoff between securing the cap during launch and unsecuring it for use. Ensuring the former would tend to complicate the later. Designing secure latches with a simple unlatching mechanism is, again, a fairly generic problem. As the engineer and computer scientist pursued this narrow task, it became clear that the most important knowledge in these deliberations were what kinds of mechanical configurations can achieve certain ends, and what accumulated experience says about the pros and cons of each option.

Despite its relative simplicity, this real-life scenario has illustrated many important points about the conceptual design of complex artifacts. This was not radical brainstorming: the engineer was not worrying about how to gather images from a space probe, or even how to protect the optics of an imaging device — he was just worrying about how to secure and then remove a lens cap. In truth, this was quite typical of conceptual design: the issues he was worrying about, the level of detail at which he was doing his reasoning, the fact that he was both exploring issues and generating initial strategic responses, the way he evaluated strategies, and the way he focused on a salient tradeoff are all characteristic of design during the conceptual stage.
Conceptual design, though a small part of the entire design process, is crucial to the overall success of many projects. As artificial intelligence researchers, we have been studying the process of conceptual design, particularly for complex artifacts such as buildings and aircraft. Our aims are both to understand the process and to develop tools to help people do the job better. Prescriptive design methodologies often miss the mark on what people are really doing when they do design. We are instead trying to offer a general description of the conceptual design process: conceptual design involves exploring for problematic issues, generating ideas about possible solutions, noting interactions that are likely to require tradeoffs, making initial strategic commitments, and often, generating a few focal commitments that can drive much of the rest of the design.

We have paid particular attention to the important role played by experience during successful conceptual design. In particular, previous design cases, whether a designer's own, or ones relayed by another designer, or ones found in a magazine or file cabinet, provide grist for a number of the decisions made during design. They raise issues that need be taken into account, suggest strategies for resolving design issues, provide illustrations of how to carry out design strategies, warn about possible interactions between multiple issues and their associated strategies, and help project the possible effects of designing something a certain way.

The practices of real designers tend to support the applicability of case-based reasoning (CBR) to design. This chapter reports on our ongoing efforts to turn this observation into the basis for a class of computer-based design aiding tools.

2. A Brief History of Case-Based Design Aids

The concept of a case-based design aid (CBDA) grew from the more general idea of a case-based decision aid. CBR (Kolodner, 1993) started with the observation that cases were often useful in solving problems. Several members of the CBR community noticed that both novices and experts seemed to seek out cases when confronted with problems. While case adaptation, merging of ideas from several cases, and case-based critiquing seemed to come naturally, these human reasoners didn't always have access to relevant cases when they could
use them. At the same time, CBR researchers found that creative case adaptation, merging, and critiquing were hard for computers.

Out of these observations came the idea of building human-computer systems for case-based reasoning: the computer aid would serve as a well stocked, well organized, and reliable external memory for past experiences; the person would be in charge of reasoning based on the available cases (Kolodner, 1991). This idea has since been developed in several domains: autoclave loading (Hennessy and Hinkle, 1992), architectural design (Domeshek and Kolodner, 1992), aircraft subsystem design (Domeshek, Herndon, Bennett, and Kolodner, 1994), user interface design (Barber, Bhatta, Goel, Jacobsen, Pearce, Penberthy, Shankar, and Stroulia, 1992), trust bank accounting (Ferguson, Bareiss, Birnbaum and Osgood, 1992), and tax consulting (Slator and Riesbeck, 1991).

A major issue in designing these systems was the content of cases and the form in which to present cases to human reasoners. Whole complex cases are too big to present at one time to a person (Pearce, Goel, Kolodner, Zimring, Sentosa, and Billington, 1992; Domeshek and Kolodner, 1991). On the other hand, every large case can be partitioned into a set of interesting and useful lesson-bearing narrative chunks. It is these chunks, sometimes called “stories” (Schank, 1991), that are at the heart of case-based aids.

Stories capture past experience, recording real world situations and analyzing their outcomes. Stories provide useful information for reasoners who are faced with similar situations, and who must make decisions about how to cope with them. Stories are most effectively told in a vivid first person voice by those who lived through, or were affected by, the situation. In some systems, stories are video segments of interviews in which experienced workers tell their most interesting “war stories” (Slator and Riesbeck, 1991; Ferguson, et. al., 1992). In our design aids, stories are text and graphic presentations that report evaluations drawn from many sources. The following is an example of a story (see also Figure 1) in the Archie-II CBDA for the conceptual phase of building design (Domeshek and Kolodner, 1992):

I consider the case of the house you live in -- not huge as cases go, but too large to present to anyone at one time. On the other hand, you might have an interesting story to tell about the design of your kitchen or the placement of your living room window or the design of the tile in your bathroom, and so on.

---

1Consider the case of the house you live in -- not huge as cases go, but too large to present to anyone at one time. On the other hand, you might have an interesting story to tell about the design of your kitchen or the placement of your living room window or the design of the tile in your bathroom, and so on.
The Roswell Library has a vaulted ceiling that gives a dramatic, spacious, and dignified feel to the interior. It makes the building appear larger on the inside than on the outside.

The layout is traditional and symmetrical; it is orderly and balanced. Yet the interior is also very contemporary and warm. Pale yellow paint on the upper walls, and gray on the lower ones, create a pleasant and inviting atmosphere. Natural light is brought in through the clerestory — a vertical window placed in a raised portion of the roof.

If you look at the clerestory you can see an 'L' louver from the edge of the overhang. This keeps harsh summer sunlight from damaging books, and it keeps the cost of cooling the library low. The placement of the louver allows sunlight in the winter to penetrate the clerestory areas. White moldings reflect the light back, and winter sunlight is thus maximized. This passive solar design complements the energy efficient design of the building, which adheres to the high standards of the Georgia Energy code.

Most case-based decision aids are further characterized by an assumption that the user is engaged in an ongoing problem-solving task or goal-directed process (e.g., a designer using a design system might be engaged in design, or in evaluating the design of another designer). Stories have the greatest value when users are ready to hear them — when users need to assess a situation similar to one analyzed in a story, when they are confronting a decision for which the story offers advice, when they have proposed a course of action that the story critiques.

A case-based decision aid and a user engage in an extended dialog: the user composes a series of situation and response descriptions, which elicit a series of stories from the system. The effect of this iterative process is to refine an understanding of the situation and to compose a well thought out response. For example, the story above might have resulted from a query about lighting, and in that context raises the issue of energy efficiency; follow-up queries might dwell on different ways of achieving energy efficiency in situations where shielded clerestory would not work.

CBDAs are an application of the basic case-based decision aiding insight to the task of design, or more specifically, to the conceptual
design of complex artifacts. This work is a natural outgrowth of combining the idea of case-based aids with a long-standing research program in case-based design (Kolodner, and Penberthy, 1990; Hinrichs, 1992; Goel, and Chandrasekaran, 1989). Archie was the first, and is the longest running, CBDA project. The original Archie system (Pearce, et. al., 1992) let users partially describe a building and in return offered documentation on similar buildings. That first version taught us many lessons about how experience needed to be packaged if it was to be accessible and useful to a designer (Domeshek and Kolodner, 1991; Goel, Kolodner, Pearce, Billington, and Zimring, 1991; Pearce, et. al., 1992).

The primary lesson of the original Archie was that records of entire designs were too large and unwieldy to be very useful to working designers (Pearce, et. al., 1992). What we had to do instead was to carve what was known about existing artifacts into smaller presentations, each focused on making a particular point. These lesson-bearing chunks became the stories of the Archie-II system. For instance, the story above focuses on the way light is handled in the main hall of the Roswell library, describing the interaction of ceiling height, coloration, window placement and sun shielding, but leaving out everything else about the design of the building.

A second consequence of experience with Archie was our reconceptualization of stories as rich multimedia presentations employing the formats and conventions natural to the designers we aimed to support. Figure 1 shows the sample story from above as it appears in the current version of Archie-II, accompanied by a photograph illustrating how the clerestories bring in light. Figure 2 shows a floor plan of the building under discussion. Whenever possible, Archie-II presents evaluative stories paired with relevant design documentation to aid story interpretation. Experience with the original Archie also led us to focus on creating systems in which designers could quickly browse through a growing collection of small lessons, as well as among different artifacts and their natural forms of documentation. The facilities we have developed for browsing will be discussed in the next section. Those facilities were shaped by consideration of how the different phases of the conceptual design process require different access pathways.
The Roswell Library has a vaulted ceiling that gives a dramatic, spacious, and dignified feel to the interior. It makes the building appear larger on the inside than on the outside.

The layout is traditional and symmetrical; it is orderly and balanced. Yet the interior is also very contemporary and warm. Pale yellow paint on the upper walls and gray on the lower ones create a pleasant and inviting atmosphere. Natural light is brought in through the clerestory—a vertical window placed in a raised portion of the roof.

If you look at the clerestory you can see an 'L' louver from the edge of the overhang. This keeps harsh summer sunlight from damaging books, and it keeps the cost of cooling the library low. The placement of the louver allows sunlight in the winter to penetrate the clerestory area. White moldings reflect the light back, and winter sunlight is thus maximized. This passive solar design complements the energy efficient design of the building, which adheres to the high standards of the Georgia Energy Code.

Figure 1. A Sample Story from the Archie-II CBDA.

Figure 2. Design Documentation for the Story in Figure 1.
Since setting off in this new direction, there have been several rounds of implementation. An initial SuperCard mock-up of Archie-II was superseded by a first Lisp implementation. That stand-alone implementation was later replaced by a version built on top of our own CBDA shell, a tool built in Macintosh Common Lisp. The shell, called Design-MUSE, is now in its second major revision and a new architectural database is being constructed to take full advantage of recently added features. Section 4 will discuss the experiences we have had using Design-MUSE to construct and run CBDAs, primarily in educational settings.

3. Browsing in a Case-Based Design Aid

This section aims to answer two primary questions: What kinds of information should be in a CBDA? How should the system's contents be made accessible? We know that people need access to cases during conceptual design and we know a system will be more helpful if it pre-digests those cases into lesson-bearing stories. What else might be needed to get the most value out of stories during conceptual design?

We note that the kind of support a designer needs is likely to shift during the course of conceptual design. Accordingly, we organize our discussion of CBDA contents and access by considering a likely browsing sequence reflecting different phases of conceptual design. We identify four phases:

1) Orientation and issue discovery;
2) Issue understanding and elaboration;
3) Issue and tradeoff exploration;
4) Proposal critiquing and evaluation.

3.1. ORIENTATION AND ISSUE DISCOVERY

When starting out on a new project, a common way to get oriented is to study similar completed projects. For instance, an architect designing a manufacturing research center told us how he had traveled the world to visit such buildings at other universities and at corporations. The point of such an initial survey is to explore broadly what the artifact must accomplish, what might be difficult or problematic, and how a design might achieve the desired ends. CBDAs are set up to provide access to design documentation based on
partial descriptions of major design goals, constraints, and environmental factors. In Archie-II, an architect designing a new public library might start by searching for libraries of an appropriate size range in suburban settings. The result would be documentation on those buildings in the system's library that come closest to fitting that description.

Figure 2 shows a sample of such building documentation displayed in Archie-II's "Design Pane." One important feature of that display is the way the CBDA highlights lessons by using the standard architectural convention of annotating design graphics. In the left margin, a short note summarizes important design features ("high ceilinged hall" and "clerestory windows") and their effects ("bright spacious feel" and "minimizing layout and energy problems"). That annotation is tied to a dot on the floor plan indicating the part of the building being discussed. Many such dots are scattered over the floor plan, each indicating that there is something noteworthy about the design — each dot represents a story to be told. Clicking on a dot or on its corresponding annotation causes the related story to be displayed. The set of dots whose annotations actually appear in the left margin at any given moment depends on the set of stories selected as relevant at that time.

In the context of the current discussion, the important point is that the user can be led from an initial undirected survey of some existing artifact to a more detailed consideration of the lessons that can be learned from that artifact. The linkage from design documentation to stories is the first step in drawing the designer into a deeper understanding of the issues they face and of design options available to them.

3.2. ISSUE UNDERSTANDING AND ELABORATION

A story describing something that worked out well or led to trouble in a previous design can serve to raise an issue, but on its own, it cannot give a full understanding of the possibilities and interactions that might arise in other situations. A full grasp of any issue usually requires some general understanding of what is at stake, what strategies are available for coping, as well as how the problems and possible solutions are likely to manifest in a variety of contexts. Accordingly, CBDA's provide not only specific stories, but also general problem and
response statements that provide context for interpreting stories. Design documentation associated with stories allows the story to be interpreted in the broader context of the design case it is embedded in; problems and responses associated with stories generalize the points of stories and help readers to recognize those points.

For instance, Figure 3 shows the story from Figure 1, but this time the screen shot includes a pair of flanking panes that display a related problem and response. The problem shown in Figure 3 points out the general desirability of having (some) natural light in a library and points out the difficulties with arranging for that light using normal windows. The response suggests a general way of arranging for light to be brought into reading areas in libraries. The story provides a concrete example of carrying out the response.

Stories, problems, and responses have symbiotic relationships. A problem statement associated with a story helps highlight the significance of the design features described in the story, while the story makes the general description in the problem statement more concrete. A story associated with a response provides a concrete illustration of how to carry out the general solution described in the response statement; the response statement points out the attributes of the story that the reader should focus on.

Because problems and responses may apply to several stories, they can serve as organizers of related stories, providing access to related similar situations. This helps a designer to explore the ins and outs of some design issue, e.g., the range of ways in which a design problem manifests itself, the different ways its solutions can be carried out, the range of situations in which some response is applicable, and so on. Together a set of stories can illustrate when a particular response strategy is appropriate and when it might fail, a variety of ways a general response might be implemented, or some of the pitfalls inherent in adopting a type of response. In addition to having several stories flesh out a response, CBDAs assume there are usually several responses available for each identified problem; in design it is rare that there is only one way to cope with an interesting problem.

---

2A response here is taken to be a general but fragmentary solution strategy.
The normal window space in a building is along the walls, but in libraries, wall space is at a premium for offices and bookstacks. In addition, internal bookstacks within internal walls that may be impenetrable to natural light.

MUSE atriums, natural screens composed of plants, and indirect light panels.

**Figure 3.** Problem/Response/Story Grouping from Archie.
Just as we led the designer from their initial survey of similar artifacts to consideration of the significant stories of those artifacts, so now we want to lead the designer to a deeper understanding of the issues raised by those stories. Annotation dots on design documents indicate the availability of lesson-bearing stories. The triangle formed of six arrows symbolically linking the "Problem" "Response" and "Story" panes indicate that the designer may traverse a network of linked presentations to explore issues in depth. When viewing a presentation in any of these three panes the user may choose from pop-up menus associated with these arrows to select related presentations in either of the other two panes.

3.3. ISSUE AND TRADEOFF EXPLORATION

Problems, for the most part, represent interesting interactions among design issues that require some kind of tradeoff for the best resolution. Through the kinds of browsing we have sketched so far, issues can be identified, interactions noted, and responses explored. But as a designer gets deeper into conceptual design they will generate their own concerns with other issues and tradeoffs. At some point, a more focused mechanism for finding lessons becomes useful. For this phase, CBDA's provide search facilities based on user-specified interests, including design issues and partial artifact specifications. Stories (and problems) are not only linked to design documents and to other lesson materials, but they are also indexed directly. The trick is to invent an indexing language in which it is possible to express many of the concerns that might arise during conceptual design.

For instance, having browsed through the set of stories clustered around the problem and responses in Figure 3, an architect would have some good ideas about ways of bringing natural light into a library while allowing for book stacks. In the course of starting to lay out study areas for visitors and work areas for employees, the architect could reasonably become concerned with how decisions about natural lighting might impact the users' sense of having private space to work in. Having arrived at this concern on her own, she needs a way to query the system about this issue.

Figure 4 shows the "Find" window from Archie-II with a query asking for problems arising from the interaction between getting natural light into work areas and the provision of adequate privacy for
Figure 4. Find Window from Archie with Query for Problems. workers. The format of queries in this window (how they appear to the user) is one of the current areas of active work. The content of such queries, however, is sufficiently settled that Figure 4 can be used as a good basis for discussion.

The first thing to note is that a query can be composed of multiple interests which may be in competition, and thus may require some tradeoff. In Figure 4, the first interest includes the old issue of getting natural light into the users' work areas; the second interest mentions the new concern about the users' sense of privacy. When the system scans its library for problems that share features with this query, it will prefer problems indexed by both issues.

The second thing to note is that design issues are construed broadly, and explicitly so. Not only are there many possible choices for the "Issue" slot of the query form, but there are separate slots in which to specify the phases of the artifact's lifecycle during which the issue might matter ("Lifecycle") and the particular class of people who might be concerned about the issue ("Stakeholder"). Thus a user can pose queries about quite specific issues, such as an owner's concern about costs during maintenance, or the neighbors' concern about safety during construction.
The third thing to note is that in addition to talking about issues (further specified by lifecycle and stakeholder), queries may focus attention on particular parts of the artifact. In Archie-II, we currently recognize two main principles for carving artifacts into parts: physical contiguity and functional relatedness. The “Space” slot provides a way for the designer to designate a physical part of a building. The “System” slot is where the designers can designate a functional subsystem. In Figure 4, both interests are focused on those spaces that are work areas; the first interest further specifies that the designer is concerned with the building’s lighting system (while the second interest implicates no subsystem in particular).

Each of the five slots of an interest specification has a defined set of fillers from which the user can choose. Users are free to specify multiple fillers for each slot or to leave slots blank. Once a query is composed, the user can request the CBDA to search. The result of a search is a new selected set of presentations in some pane. The left part of Figure 5 shows the “Problem” pane following a search using the query in Figure 4; the first of four partially matching problems is on display. The right part of Figure 5 shows the index that was assigned to the problem displayed on the left. Note that this index is similar to, but richer than, the search query. The index includes many more descriptive terms and sometimes uses slightly different terms than the query. The system’s nearest-neighbor partial matching algorithm can tolerate such differences, producing a graded sense of similarity between descriptions.

3.4. PROPOSAL CRITIQUING AND EVALUATION

At some point in the conceptual design process, users will have developed their own sketchy proposals, and at that point the system ought to be able to offer more focused critiques. This is an area where the current version of Archie-II is weak. To better support critiquing and evaluation, the system would have to allow for more detailed descriptions of partial design solutions than it currently does. This is also an area of active development in the ongoing Archie project.
4. Experience with Design-MUSE and CBDAs

The first version of Archie-II, built in reaction to the original Archie, was demonstrated widely and aroused interest in the Advanced Design Division of Lockheed Aeronautical Systems Corporation (LASC). A small project was launched to build a CBDA supporting conceptual design of aircraft subsystems. The resulting MIDAS system focused on lessons about design tradeoffs in the initial design of aircraft hydraulic systems (Domeshek, et. al., 1994). In about six months, with coaching from our group at Georgia Tech, a junior LASC engineer gathered and documented three dozen stories plus related problems and responses, developed an indexing system, and constructed a prototype CBDA.

What made the MIDAS project possible was that while the engineer prepared the specific information on aircraft and hydraulic system design, we were generalizing the capabilities originally developed for Archie-II into a generic CBDA shell supporting creation of new CBDAs for any kind of complex conceptual design. An important goal of the shell effort was to devise a simple interface affording domain experts the capability of growing their own CBDA libraries.

Figure 5. First Result of Figure 4 Problem Search (with matching index).
MIDAS was first constructed on top of our initial shell prototype. As that prototype evolved into the shell we now call Design-MUSE, we continued to use MIDAS as a sample case library; MIDAS now runs in version 2 of Design-MUSE (as illustrated in Figure 6).

We learned several interesting things from the MIDAS project, not the least of which was how some of the ideas initially developed in Archie transferred to another design domain. What first attracted LASC's interest was the understanding that conceptual design was a critical process, and one that could benefit from easy access to multimedia presentations of lesson-bearing stories evaluating previous designs. LASC also liked the mixture of stories with more concrete design documentation and more general discussions of issues and tradeoffs, and they appreciated the variety of access paths provided by linking these various materials.

When it came to the indexing framework, we found that the five dimensions used in Archie-II transferred directly to the aircraft domain, though the fillers did not. Table 1 shows the indexing vocabulary used in MIDAS (except that the "system" dimension, which for this prototype had only the single item "hydraulic," has been omitted). The indexing vocabulary for Archie-II is too extensive to list in this paper, and is also under active revision. The column listing MIDAS's "components" corresponds to Archie-II's "spaces" dimension, but obviously there is no overlap in the terms provided. Likewise, the "issues" dimension shares little between the two systems other than the inclusion of similar classes of issues (e.g. parameters, pitfalls, requirements). The "life cycle" and "stakeholder" dimensions showed the greatest commonalty between systems, though even here there are variations, and we consider many more stakeholders in the case of buildings.

There were also three less expected results. The first we probably should have expected — we were confronted with the importance of the social dimension when it comes to knowledge acquisition. We had imagined the bulk of our materials would come from interviewing a senior design engineer, but unfortunately, we found this expert to be quite reticent about sharing his "war stories." Whether due to the security-conscious culture at LASC, a misconception of what we were asking for, a low valuation of his anecdotal experiences, or a high valuation of his special expertise, our expert told few stories.
When incorporating a high flow rate pump into a hydraulic system design, vibrations from pump cavitation should be considered.

Vibrations caused by cavitation of a high flow rate pump (40 gpm) caused line and fitting failure that led to loss of an aircraft. The problem was resolved by adding an air-oil accumulator to the system, which damped out the pump pulsations. Adding a fluid damper or a quickie (flexible, u-shaped) tube can also significantly decrease the air contamination in the fluid and, in turn, reduce system vibrations from high flow rate pumps. Solutions that attempted to increase the structural integrity with welded lines and increased wall thickness still failed from the pump vibration.

The most effective way to damp the vibration produced by cavitation of a high flow rate pump is to incorporate an air-oil accumulator which acts as a fluid damper and reduces the air contamination.

Theory of Quickie Tube

$$A_1 = A_2 + A_3$$

Excess pressure at junction of 2, 3, and 4 is zero, so a wave or pressure drop traveling from left to right cannot be transmitted to line 4.
## TABLE 1. Indexing Vocabulary from MIDAS.

<table>
<thead>
<tr>
<th>Components</th>
<th>Issues</th>
<th>Life Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accumulator</td>
<td>Parameters</td>
<td>Design</td>
</tr>
<tr>
<td>Actuator</td>
<td>Electric</td>
<td>Fabrication</td>
</tr>
<tr>
<td>Controller</td>
<td>Flow-rate</td>
<td>Assembly</td>
</tr>
<tr>
<td>Converter</td>
<td>Material</td>
<td>Test</td>
</tr>
<tr>
<td>Filter</td>
<td>Power</td>
<td>Transport</td>
</tr>
<tr>
<td>Fitting</td>
<td>Pressure</td>
<td>Use</td>
</tr>
<tr>
<td>Fluid</td>
<td>Volume</td>
<td>Maintenance</td>
</tr>
<tr>
<td>Heat-exchanger</td>
<td>Weight</td>
<td>Repair</td>
</tr>
<tr>
<td>Landing-Gear</td>
<td>Pitfalls</td>
<td>Disposal</td>
</tr>
<tr>
<td>Manifold</td>
<td>Contamination</td>
<td></td>
</tr>
<tr>
<td>Motor</td>
<td>Cavitation</td>
<td></td>
</tr>
<tr>
<td>Pressure-transducer</td>
<td>Temperature/Heating</td>
<td>Stakeholder</td>
</tr>
<tr>
<td>Pump</td>
<td>Vibration/Acoustics</td>
<td>Builder</td>
</tr>
<tr>
<td>Reservoir</td>
<td>Leakage</td>
<td>Customer</td>
</tr>
<tr>
<td>Sensor</td>
<td>Requirements</td>
<td>Designer</td>
</tr>
<tr>
<td>Valve</td>
<td>Effectiveness</td>
<td>Mechanic</td>
</tr>
<tr>
<td>Wiring</td>
<td>Efficiency</td>
<td>Pilot</td>
</tr>
<tr>
<td></td>
<td>Installation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Range/Payload</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Safety</td>
<td></td>
</tr>
<tr>
<td><strong>Life Cycle</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Affordability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Producibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deployability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supportability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintainability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Survivability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flexibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vehicle-growth</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Life-cycle-cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Environmental-impact</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The second interesting point, however, was that this same expert seemed delighted to share documents from his personal library, and partly as a result of his attitudes we learned that there is much useful but normally inaccessible information in documents that could be organized by a CBDA. For instance, there are many studies evaluating trials of new technology in existing aircraft, and reports are often written documenting system failures. These are rich veins of experiential knowledge.

The third interesting discovery was that a CBDA library would not have to be tremendously large to be useful. Instead of the thousands of stories we had imagined, the Lockheed engineers seemed convinced that a system with as few as a hundred stories might be useful in a narrow domain, and with just several hundreds would be quite useful over much of initial utility subsystem design. As a trial of our small prototype system we compared the results of a published design tradeoff study examining alternate hydraulic system operating pressures with information gathered from searching in MIDAS; MIDAS provided quick access to the equivalent information and actually did a better job of raising issues and citing precedents.

As a final benefit, we also learned much about the usability of some of our interface concepts as day-by-day feedback from our Lockheed collaborators helped shape many of the features of the evolving CBDA shell. Everything from screen layouts, to label choices, and icon designs was criticized. The range of linkages among presentations was simplified. More support was developed for entering, linking, and indexing materials.

4.1. ADDITIONAL EXPERIENCES BUILDING CBDAS

The MIDAS project ended in August of 1993. By then, however, we were well on our way into the first major revision of the CBDA shell. By January of 1994 Design-MUSE Version 1 was available, and we put it to use in a graduate course on case-based reasoning run by Professor Kolodner (Domeshek and Kolodner, 1994). The goal was to gain more experience with supporting CBDA construction by people whose primary expertise is in domain content rather than indexing techniques. During a 10 week quarter small groups within the class used Design-MUSE to produce five new CBDAs:
• **Next-Archie**: A revised-extended version of the *Archie* CBDA for building design, aimed especially at student architects. (Design education)

• **Composites**: An advisor to help with decisions about replacing metal parts of aircraft with parts made from composite materials. (Re-design)

• **DSP-Aid**: A guide to formulating design problems in the "Decision Support Problem" (DSP) methodology so they are suitable for analysis using DSP software tools. (Design process)

• **GT-SCARES**: An on-line trouble log to help in diagnosing and coping with satellite failures. (Diagnosis and debugging)

• **Dream-Car**: An advisor for consumers considering purchase of a new car. (Situation assessment)

As with *MIDAS*, we learned some interesting things from having domain savvy AI novices build case libraries using our shell. First of all, these systems are not all CBDAs as we had originally conceived them. While all aim to support decision-making with respect to complex artifacts, they vary quite a bit in terms of what decisions they support. Some have little to do with design, per-se.

Secondly, this exercise generated more divergence in the indexing frameworks. When we gave each student group their copy of *Design-MUSE*, we provided an initial index framework with five dimensions: *part*, *system*, *issue*, *stakeholder*, and *life cycle*. We also supplied a basic set of primitives for the fillers of the stakeholder and life cycle dimensions. None of the groups stuck with this initial framework. This was not terribly surprising in the case of systems such as *GT-SCARES* or *Dream-Car* which were not primarily concerned with aiding designers. It is more interesting in the cases of systems like *Composites* and *DSP-Aid*. (*Next-Archie* stayed much closer to the initial framework). The variety raises questions about how we can most effectively generalize our design-aiding work and build support into our shell so it gets easier for domain experts to build new CBDAs.

About the same time as this first class experiment, another project at Georgia Tech's Center for Human-Machine Systems Research began to test some of the core ideas of CBDAs, but without directly using *Design-MUSE*. Jennifer Ockerman's recently completed work (Ockerman, 1995) describes the *CMS-Browser*, a system she built to facilitate design reuse by NASA software contractors. She constructed a repository for analyses of several independently developed satellite
command management systems. Those analyses were broken up into focused discussion of particular parts and issues, and cross linked so as to facilitate comparison across projects. Evaluation tests at NASA suggested the system would be effective at helping designers recognize commonalities across software systems at a conceptual level.

4.2. BUILDING EDUCATIONAL APPLICATIONS OF CBDAS

Development of Design-MUSE continued, and related CBDA projects began to focus more on educational applications. During the winter quarter of 1995 Version 2 of Design-MUSE was exercised, again in Professor Kolodner’s class. This time, three groups of students worked to produce three parts of a single case library on the topic of sustainable technology and development. One group focused on cases of industrial pollution, another on sustainable development and resource management, and the third on industrial accidents. The merger of these three pieces by Hari Narayanan produced Susie (SUStainable technology Interactive Education). The faculty who run Georgia Tech’s three quarter sequence on sustainable technology are currently considering whether to use Susie as a seed library for their students to learn from, and then extend.

The best example of work towards educational CBDA applications is CLIDE (Case Library for Industrial Design Education). Initially being assembled in the fall of 1995, CLIDE is intended to support two new courses on product design in the Industrial Design Program at Georgia Tech. The goal is that students in the first course (focusing on Design Case Studies) will extend CLIDE each quarter; then beginning in 1996 students in the second course (dealing with Collaborative Product Design) will make use of the resulting case library. Both courses will be offered every quarter.

4.3. EXPERIENCE USING CBDAS IN EDUCATION

The first attempt at educational use of a CBDA (for end-user browsing as opposed to system construction) occurred in the Spring quarter of 1994 in the Georgia Tech College of Architecture. A version of Archie-II stocked with information about libraries and courthouses was made available to students in two studio sections that were involved in a library design competition. One of the two sections
was composed of sophomore undergraduates, the other of 2nd-year masters students. We worked with the students only during the later half of the quarter, spending about two weeks with the undergraduates (just before they submitted their designs), and then two more weeks with the graduates (just after they had submitted theirs).

After group introductions to the system, each of our dozen students spent one or two hour-long sessions working through exercises we had designed. The first session's exercise involved critiquing an existing library on which we provided documentation; the second session, which was held only for the graduate students, involved critiquing their own designs. A couple of students also came back at other times simply to browse the system for their own interest.

Our pedagogical goals included familiarizing students with a range of past designs, and reinforcing the habit of critically assessing design by consistently linking actual outcomes to relevant design intentions. We wanted to start evaluating our CBDAs both with respect to their use in teaching, and with regard to their general usability, and the usefulness of their information content for conceptual design. The results were a bit mixed but still encouraging. All the students liked the idea of the system, but the graduate students appreciated the actual content far more than the undergraduates. Interestingly, the undergraduate studio instructor judged that those students who used the system had gotten a lot out of their limited exposure, even though there was no way to directly trace any details of their designs to their use of the system. As for instilling a critical attitude in which designs are habitually related to intentions, there is no evidence as to whether or not any of the students absorbed that lesson.

Overall, the first studio experience was positive enough that Archie-II has been undergoing continued development in preparation for another, more substantial, classroom trial now tentatively scheduled for the Winter 1996 quarter. With the help of one of Georgia Tech's studio instructors, a new corpus of materials bearing on the design of tall buildings is being assembled. As with the other educational applications outlined earlier, our goal is to have students both contribute to the case library, and use it to support their own design work. We believe that both exercises should have significant pedagogical value.
5. Related Work

Using case-based reasoning in design (Hinrichs, 1992; Sycara, Navin Chandra, Guttal, Koning, and Narashimhan, 1991; Hua, Smith, Faltings, Shih, and Schmitt, 1992), using cases to support design (Domeshek and Kolodner, 1994; Oxman, 1993; Nakakoji, 1993; Voss, et al, 1993), or simply capturing design cases in process (Conklin and Begeman, 1989) have all become popular areas for research. There are now a variety of projects that have looked at ways to use cases to aid designers (including many in the domain of architecture). Surveying those projects reveals a set of issues where different researchers have taken different stances. Here we will focus on the following issues:

1) What kind of design is supported?
2) What kind of users are assumed?
3) What kind of case materials are provided?
4) How are case materials made available to the designer?
5) What does the program let designers do with case materials?

5.1. WHAT KIND OF DESIGN IS SUPPORTED?

Most work on AI in design has focused on detailed design, often in simplified circumstances. This holds true for work on CBR in design, and to some extent, even for work on case-based design aiding. For instance, Nakakoji’s main domain was kitchen configuration using a limited pallet of possible kitchen components. Interestingly, the FABEL project, based on the ARMILLA model, aims to document the entire history of complex buildings at all levels of detail. In contrast, CBDAs are aimed squarely at conceptual design of complex artifacts. In this, as in many other ways, our work is most similar to Oxman’s information repository.

5.2. WHAT KIND OF USERS ARE ASSUMED?

We imagine two kinds of users in our work: professional designers and students. To satisfy both audiences, we must address the issue of how a case library might differ when targeted for students and when aimed at professionals. Our assumption has been that student designers are designers, though they are still learning, and therefore much of the
help that is useful to professionals is useful to students. We also assume that professional designers, no matter how expert, learn from seeing critiques and analysis of other projects. Thus we believe the form and content of case libraries for these two distinct user groups might not be terribly different.

It is true that some lessons appropriate for a student would be too elementary for a professional (and likewise, some cases of interest to a particular practitioner would be too idiosyncratic for a student). Nonetheless, many of the design points illustrated by the successes and failures of existing buildings are of interest to both groups of users. Since users control what issues they focus on, a representative and well-documented case library can serve both user groups well. On the other hand, it might be appropriate to supplement a student system with basic information about standards and guidelines — resources that a professional might access more rarely. One thing we must remember is that a case library is but one resource available to designers as they design; it need not provide every shred of information a student or professional designer might need.

To pursue the notion of difference further, however, we can imagine more focused CBDAs diverging for students and professionals. A system for students might focus on a famous designer, an important design style, or on a group of precedent-setting buildings. Oxman has pointed out that in architecture some precedent-setting buildings recognized as breaking boundaries or setting standards are important for working designers to be familiar with. Since these buildings are well studied and documented, CBDA's could be used to present them to students. At the other extreme, a CBDA might be built for a particular narrow group of professionals — perhaps the members of one design firm — reflecting the idiosyncratic experience of their particular design practice.

There are several more interesting ways in which a CBDA might be tuned for student use. First, student designers might not be familiar enough with the vocabulary of a domain to be able to use an indexing system based on the categories and shared concepts of experienced practitioners. A student-oriented system might focus on providing appropriate “scaffolding” (Wood, Bruner, and Ross, 1975; Collins, Brown, and Newman, 1989) to introduce students to the vocabulary and concepts of the domain. Similarly, one can assume that students, initially unfamiliar with carrying out conceptual design, might need
help in deciding what to investigate at any time and in recognizing the most important places to focus attention during their deliberations. The guidance provided here should be such that it helps students learn to do design. So far as we know, nobody is working directly on the problem of scaffolding for student designers.

5.3. WHAT KIND OF CASE MATERIALS ARE PROVIDED?
No matter the source and no matter the audience, we believe the most useful case libraries for conceptual design will emphasize certain kinds of case materials — in particular, lessons and documentation, augmented by generalized problems and responses. We suggest avoiding masses of unanalyzed design data. Much detailed information on artifacts is irrelevant to conceptual design lessons and can be excluded from such a system. Likewise, on-line catalogs of parts for configuration are not needed to support conceptual design.

The bias towards conceptual design differentiates CBDA’s from other design aids that use cases. Were CBDA’s to be broadened to cover other phases of design (and this might be an interesting direction in which to move), materials needed for those other phases (e.g., extensive detailed design documentation, extensive descriptions of component parts) would need to be integrated into the presentation framework as well.

5.4. HOW ARE CASE MATERIALS MADE AVAILABLE?
CBDA’s provide several means of navigating or browsing a case library, and we have discussed these extensively in Section 3, justifying the particular means we provide by appeal to the process of conceptual design. The kinds of access provided depends not only on the demands of conceptual design, but also on the availability of certain classes of presentations. Most other systems do not provide such a rich variety of materials, and thus cannot provide linkages, for instance, between documentation and evaluation, or between general and specific information. Except in systems where focused presentations address particular issues and interactions, the kind of interest-based indexing promoted by Design-MUSE is not a useful technique.
Irrespective of the exact content of indexes in a system, the system might take a stronger hand in forming queries. A proposal due to Mary Lou Maher (personal communication) suggests a way in which the accessibility of indexed information might be improved through letting the system itself suggest components of search cues when a partially formed query indicates the user is interested in related areas.

Ultimately, we want to see CBDA's integrated with work environments, somewhat like that provided by Janus (Nakakoji, 1993), but with greater flexibility in response to the demands of conceptual design of complex artifacts. Such a system could watch a designer's work in progress and offer appropriate advice and information based on evolving design proposals. Nakakoji described "case delivery" in the restricted case of detailed kitchen design. Further work in this direction requires, at the very least, a more completely fleshed out on-line conceptual design environment. Initial steps in this direction are described in the final section below.

5.5. WHAT CAN BE DONE WITH CASE MATERIALS?

The final issue is what a designer can do with retrieved case materials. Even in as complex a domain as architecture, most case-based design systems try to make pieces of retrieved cases fit into a new design (Hua, et. al., 1992; Nakakoji, 1993). Because these systems are aimed toward more detailed design, they emphasize a direct kind of reuse that is usually inappropriate during conceptual design. Instead of expecting a designer to reuse a particular part of a building configuration (aided by parametric adaptation to fit new circumstances), we expect our designers to use cases to learn a lesson that can then be applied to the novel circumstances of the new situation. This might require radical reconfiguration when combined with all the other ideas and lessons picked up during browsing.

We do not yet know exactly what kind of aid we should provide to designers to help them with this merging process. The closest related work would probably be systems that focus on design rationale capture or design space exploration (Conklin and Begeman, 1989). While these systems do not emphasize access to lessons, they do suggest ways to help a designer organize the kinds of insights such lessons might impart.
6. Future Directions

This chapter has brought the story of case-based design aids up to the present. We have built a series of trial CBDAs, we have developed a domain-independent CBDA shell, and we have conducted some preliminary trials of these systems looking both at the construction of CBDAs and at their use in educational design settings. As the work has progressed, our vision has evolved. From our current vantage point, we can pick out several features that are central to the structure of CBDAs:

1) CBDAs are intended to support designers working on the conceptual design of a complex artifact;
2) The primary support a CBDA offers is easy access to prior experience;
3) Huge design cases are broken up into small lesson-bearing stories;
4) Stories are linked to general discussions of design issues and design strategies, emphasizing the tradeoffs inherent in most complex design situations;
5) The issues highlighted in a CBDA encompass the entire lifecycle of an artifact and represent concerns of all stakeholders;
6) All presentations can make appropriate use of whatever media forms are natural to working designers;
7) The presentations are organized so they are accessible in ways that fit the needs of the different phases of conceptual design.

The major task before us is to demonstrate the utility of CBDAs on a larger scale. The pending exercises — continued stocking and student use of CBDAs for architectural design, sustainable technology, and product design — are steps in this direction. Further progress in a corporate environment such as LASC appears to require major commitment of resources, not only to gather, prepare, and enter materials, but also to support reimplementation of the system to cope with real world requirements such as multiplatform portability, efficient centralized data storage, and standardized presentation
formats. For many smaller companies, it might be sufficient to simply rebuild the system for Windows PCs.\(^3\)

Our experiences so far have turned up several needed improvements; we are currently working on some of these. One deficiency we intend to address is the way that the indexing system is presented to the user. Currently, the query interface is almost totally transparent: the menu-driven fill-in-the-form interface (as shown in Figure 4) reflects exactly the structures that are used internally (by the system) for indexing; the hierarchical menus of filler choices reveal much of the conceptual space of index terms. The result is not as comprehensible or as usable as might be a system that buffered the user a bit more, presenting choices in a way more tailored to the mature practitioner's understanding of the domain. Another deficiency is in the expressiveness of the indexing system. While we have worked hard on capturing the cross-domain dimensions of story indexes, our story indexes do not allow specification of the larger design a story is part of. Thus, one can ask the system for stories about lighting but not for stories about lighting in thick buildings.\(^4\)

A related issue is the extent to which we have captured the appropriate generalizations for easing indexing across domains and for a variety of conceptual design tasks. Our experience with the student-built CBDAs indicates we have not yet crisply characterized what should be in an index. We also know that we are not yet doing a good job of supporting critiquing of partially-specified user design proposals.

One way in which we hope to improve design proposal critiquing is by building up the design environment of which a CBDA is supposed to be a part. CBDAs already include a notebook facility in which designers can clip and save interesting presentations they find while browsing, and to which they can add their own annotations. That notebook should grow into a more fully fleshed out analog of a designers notebook — a workspace that allows (and ultimately

---

\(^3\)Work on indexing video clips in the VID project (Gordon and Domeshek, 1995), a tangentially related effort at Northwestern University's Institute for the Learning Sciences, might ultimately facilitate the move of CBDAs to the popular Windows platform by establishing basic data handling infrastructure.

\(^4\)A thick building is one that is so deep or wide that lighting from the outside cannot be used to light the center. Large square buildings are "thick."
facilitates) sketching, calculating, and perhaps even drafting or modeling. An idea of what we intend can be gleaned from initial work directed at linking Archie-II to a sketch pad that recognizes some aspects of freehand drawings (Gross, Zimring, and Do, 1994). In this prototype, bubble diagrams of building floor plans sketched by a designer serve as queries to the CBDA library, which retrieves designs with similar structure (and then provides access to their evaluative stories). This is a direction we hope to expand significantly in the future.

Our emphasis on educational applications of CBDAs has increased the importance of providing appropriate scaffolding for student designers. We expect to pursue both of the directions suggested earlier: helping students learn the vocabulary of the new design domain (reflected, for instance, in the CBDA’s indexing system), and helping students through the conceptual design process itself (in part by focusing them on relevant information that will help them make progress).

One final extension of CBDA capabilities is also being pursued: efforts are under way at Georgia Tech to embed a CBDA information resource in a distributed collaboration environment. Design of complex artifacts is a process that necessarily requires teamwork over an extended period of time. The CaMile environment (Guzdial, Rappin, and Carlson, 1995) was designed to support structured group discussions that can make reference to documents prepared in any of a variety of normal microcomputer applications. A particularly useful class of documentation to reference during collaborative design discussion is the sort of case material available in a CBDA. Information on existing artifacts can be used to introduce alternatives, cited to justify design decisions, or referenced to buttress arguments during ongoing design debates. The Design-MUSE/CaMile environment allows cases to be referenced during collaboration; it also allows discussions about cases to be accessed while perusing the case library. Finally, the combination system provides the opportunity to provide definitions and illustrations of design vocabulary and context to novice users as part of an integrated system.

The future of case-based support for design seems bright. We have had much success with our experimental systems and generated interest from a variety of designers and design organizations. The ongoing construction and use exercises will expand both our case
libraries and our experience with using CBDAs. Initial integration projects already under way will tie CBDAs to design tools and collaboration environments. Refinement of Design-MUSE continues, reflecting our accumulating experience of system use and our evolving understanding of what is common in conceptual design across domains. In short, we believe we can look forward to progress on a variety of fronts that will lead both to more usable systems and a better understanding of the place of case-based reasoning in design.

Acknowledgments

Thanks to Craig Zimring, our constant collaborator throughout the Archie project, and to Hari Narayanan who has taken up the reins of the project this last year, helping to create Susie, and CLIDE. Thanks also to Ashok Goel, Michel Conan, Dale Durfee, Richard Billington, Anna Zacherl, Vijaya Narayanan, Marin Simina, Ellen Do, Osman Ataman, Husam Khalil, Ali Malkawi, Ameen Farooq, Marcia Herndon, and Andrew Bennett, who have all contributed to or influenced Archie and MIDAS. This work has been supported in part by the Defense Advanced Research Projects Agency, monitored by ONR under contract N00014-91-J-4092, by the Lockheed Aeronautical Systems Company, and by EduTech. All views expressed are those of the authors.

References


A Case-Based Design Aid for Architecture

ERIC A. DOMESHEK and JANET L. KOLODNER

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
fax: 404-853-9378
domeshek@cc.gatech.edu
kolodner@cc.gatech.edu

March 4, 1992

Abstract. This paper summarizes the current status of a project to construct a design aiding system for architects. The Archie-II system is an application of case-based reasoning techniques to the task of assisting human designers. The focus on design aiding, the choice of case-based techniques, and the resulting specification of a case browsing system are reviewed and justified in the first section. The balance of the paper then focuses on the ways in which design cases can be carved up for presentation to designers and how the resulting pieces can be indexed and organized so as to make them available at appropriate times in the design process.

1. Conceptual Design of a Conceptual Design Assistant

The value of rapid, effective, and creative design is becoming ever more apparent in today's global economy, hence the growing emphasis on applying computers to design. Unfortunately, much as we might wish otherwise, we do not yet have an algorithm for design. Researchers in the AI and Design community can take that lack either as a challenge or as a constraint. Those interested in improving the design process in the short run had better take it as a constraint. As such, it suggests that any useful design system of the near future will fall at the low end of the intelligence spectrum — it will more likely be a tool supporting current practitioners in their existing practices, than a design agent autonomously carrying out pieces of the design task on its own.

Taking this view, the question we want to ask is not "how can a computer generate designs?" but rather, "how can a computer help a designer?" That, in turn, leads us to ask a series of questions: What do designers currently do? Which of their tasks are currently problematic? Which might be done better with computer assistance? Which tasks provide a base onto which new processes might be grafted? What we are looking for is places where computers' abilities can address designers' needs and complement designers' abilities.

1.1. Conceptual Design

Conceptual design, as the first step in the design process, is in some sense the most influential. Commitments made there — determinations of what the problem is and what the broad solution should look like — have the largest effects and are the hardest to undo later. A mediocre or lousy concept can be carried through
skillfully, but it will never equal a similarly executed design based on a clear and clever concept. If we want to improve design, then conceptual design seems like an area with the potential for a high payoff.

But surely, if we lack an algorithm for design, then conceptual design must be one of the more problematic aspects. Conceptual design is a squishy sort of task, mired hopelessly in the open-ended real world — a task without a clear specification because part of the problem to be solved is identifying the problem. Conceptual design seems a realm for mysterious qualities like insight, inspiration, and creativity. This probably accounts for why so little work in AI and Design has been addressed to the problem of conceptual design.

On the other hand, if we are not trying to automate design, but rather to help designers, perhaps there is reason for hope. A system that supports designers’ creativity need not itself be creative. We must first consider what sort of assistance would be useful, and for that, we can look at how conceptual design is currently done. While we want a model of the conceptual design process to guide us, as long as we are not proposing that the system do design, it need not be a completely specified operational model; a loose qualitative model may tell us enough to get on the right track.

Looking at designers in the early, conceptual, phase of design, one thing is clear: instead of trying to work in a vacuum, designers spend much of their time thinking about existing designs, reviewing the literature, pouring over formal and informal documentation of earlier works. This is true in a field like architecture where architects delight in touring interesting buildings, browse magazines like Architectural Record, and often pull old blueprints from the company files. It is also true in fields where designers are, in fact, working on something radically new; for example, the next generation of nuclear reactors — lessons learned from past experiences help in focusing the designer on issues of importance.

One way we might hope to assist conceptual design is to make it easier for designers to find, browse, and learn from records of relevant old designs. The combination of a cognitive model describing how people can benefit from past experiences and a technology for finding and presenting such experiences could form the basis for computer systems that offer designers a new and valuable sort of assistance with their conceptual design problems.

1.2. CASE-BASED REASONING

The rationale for the research program reported here is that we have such a combination of cognitive model and technology. Case-based reasoning (CBR) (Schank, 1982; Kolodner, Simpson & Sycara, 1985; Hammond, 1989) is a paradigm within Artificial Intelligence based on a memory-centered cognitive model; the basic idea is that people are good at figuring out what to do in new situations largely because they are able to remember and adapt things they did (or saw others do) in similar previous situations. In addition to applications emphasizing reasoning tasks like
planning (Hammond, 1989; Alterman, 1988), scheduling (Mark, 1989), decision-
making (Bain, 1986), diagnosis (Hunter, 1989; Bareiss, 1989; Koton, 1988), expla-
nation (Kass, et al., 1986; Ram, 1989), argumentation (Ashley & Rissland, 1987;
Simpson, 1985; Sycara, 1987), and even design (Hinrichs, 1991, Goel, 1989). CBR
is being developed into a technology for building systems that assist human users
by presenting them with useful information chosen from organized memories of
past experiences (Kolodner, 1991; Domeshek, 1992; Bareiss, Ferguson & Fano,

In the course of developing the many experimental systems cited above, a wide
range of roles have been identified for memories of particular past experiences:
a past case might suggest one course of action or warn against another; it might
contribute to the evaluation or detailed adaptation of a possible response; while
analyzing a situation, it might focus attention on particular questions or suggest
specific answers. The problem is that getting a system to apply a case in any of these
varied ways requires a large amount of idiosyncratic real world knowledge. While
this remains a major roadblock preventing widespread development of autonomous
reasoning systems, it does not prevent cases from being usefully applied by a human
user who can be counted on to possess and apply sufficient common sense. This
is the realization that has driven some researchers to focus on the technology of
case retrieval and presentation, building systems to aid humans in their decision
making (Kolodner, 1991; Barber, J., Bhatta, S., Goel, A. Jacobson, M., Pearce, M.,

A case-based decision aid works roughly like this. The user describes his/her
problem to the system. The system recalls cases that are similar and presents them
to the user. The user makes decisions about the solution in the new situation based
on those cases, updates the problem and solution specifications, and repeats the
process. Cases recalled by the system can help the user by pointing out suggestions
for solutions, mistakes made in old cases, and results of carrying out old solutions
in the world. These systems tend to look like clever hypermedia systems; the
cleverness lies in how the information is organized and how it is presented. The
aim is to reflect the cognitive needs and abilities of human users engaged in a
particular task.

1.3. ARCHIE-II: A CASE-BASED DESIGN AID FOR ARCHITECTURE

The AI lab at Georgia Tech's College of Computing has been at the forefront of
research in CBR in general, and the application of CBR to design in particular. Work
on three case-based design aiding systems is currently in progress. Two of those
projects are attacking the problem of design in the domain of architecture. Ask-
Archie and Archie-II are both descended from an earlier system known as Archie
(Goel, Kolodner, Pearce, Billington, and Zimring, 1991). This paper describes
work on the Archie-II system.

Figures 1 and 2 give some idea of what the current version of Archie-II looks
Fig. 1. Archie-II Sample Browsing Screen

Fig. 2. Archie-II Sample Story Screen
like; in these figures the system is displaying information about the Bristol County Courthouse in New Bedford MA (drawn from, Building Diagnostics Inc., 1988). One key thing to notice in figure 1 is that the system makes it possible for users to browse over architectural graphics (in this case building plans) just as they normally would in current practice; we have found this mimicking of familiar media to be a powerful means of communication with architects. In addition, note that, just as on standard blueprints, the graphic images may be supplemented by textual annotations offering more detailed explanations of aspects of the figure: in Archie-II these annotation appear in boxes around the edge of the graphic, and also serve as moseable buttons providing access to still more information about the plan.

Figure 2 shows what the screen looks like after clicking on the annotation box "Concession stand is source of noise, disrupting court proceedings." The browsing controls filling the upper third of figure 1 have been replaced, in figure 2, by a box of text telling in more detail the story behind the chosen annotation. Associated with the story box are a new set of buttons offering the user a chance to look at still more information about this aspect of the building’s design. By choosing to examine the design responses related to this story, users can learn about possible fixes for the problem introduced by the concession. By choosing to examine the design guidelines related to this story, users can learn about the goals operative in this situation and plans for achieving those goals; from there, they may eventually be led to consider other stories (perhaps drawn from other buildings), that relate to the same guidelines, illustrating other successes and failures related to the initial story shown here.

1.4. Conceptual Design of a Paper on Conceptual Design Aiding

This first section has been concerned with characterizing the design problem, focusing on the conceptual design phase, justifying a case-based approach to assisting designers with this phase, and giving some idea of what such a system would look like. The rest of this paper is broken into two major sections. In Section 2 we focus on how design cases can be carved up into useful chunks; while in the context of most AI systems this would be considered as a problem in representation, in the context of a browsing system like Archie-II figuring out what is important in a case and what clusters of facts are related becomes largely a problem of presentation. Section 3 discusses how the presentation chunks identified in Section 2 are organized for browsing and search; this discussion includes some memory organization and some representational content. Section 4 closes the paper with a short summary of where we currently stand in the implementation of Archie-II and places this system in the context of a larger research program on design involving the development of other case-based aiding systems.
2. Presentations

Given that Archie-II is to serve as a case browser, emphasizing presentation of information to users over adaptation or application of past solutions by machine, the system's memory organization and contents are determined by what would make sense to present to a user. Our notion of browsing assumes there is a lot of information in the system and that the user will need to sort through it quickly, skipping much that is irrelevant to their current concerns. To be useful, a case library ought to contain records of many different designs; furthermore, each design case can potentially contain a large amount of information. The need to skim will hold both at the level of cases (which for Archie-II means building projects) and within cases.

In fact, Archie-II's system of presentations is primarily an attempt to facilitate browsing at a grain size smaller than entire design cases. Whole building designs are too large and complex to directly help an architect with most design decisions. To figure out what sorts of case chunks would be useful to designers, we have to ask and answer several questions: What design tasks are case materials supposed to help with? How is old case information supposed to aid in those tasks? What sorts of information make useful contributions to necessary tasks?

We have set our sights on supporting designers in the following four tasks:

- problem comprehension;
- solution brainstorming;
- goal-directed composition of high-level solutions;
- goal-directed filling in of details.

Together these account for much of what must be accomplished at the level of conceptual design. As for the question of how case materials are supposed to help with these tasks, we turn to prior CBR research on design which suggests that memories of past experience can make the following sorts of contributions:

- proposing solutions;
- identifying pitfalls and opportunities;
- suggesting evaluation criteria.

Information chunks that can support these tasks and provide these services cluster design features with an evaluation of how they perform with respect to goals that commonly arise in the design process. In other words, useful chunks present how some design goal was or could be implemented and the effect of that implementation. In presenting these chunks to users, the presentation needs to mention those facts about a piece of a design that affect some goal (or small set of related goals) along with the goal (or goal set) that is being addressed and that renders those facts relevant.

Based on this analysis, we have identified three classes of chunks worth presenting to designers; two kinds come directly from cases, the third kind generalizes
across cases. Stories are exactly the sorts of goal-focused evaluative case descrip-
tions sketched above; design stories are intended to teach lessons by example,
and the lessons they teach can be about attempted solutions that worked or failed,
situations where goals may or may not be relevant, ways of judging how a design is
likely to fare with respect to some goals. The second class of chunks, loosely titled
documentation, presents information organized by means other than goals; in par-
ticular, design documentation clusters information according to decompositions
in terms of structural components and functional systems. Note that stories also often
focus on particular structural or functional parts of designs, but documentation is
distinguished by its lack of a particular evaluative “point;” it is provided for those
times when the designer really wants access to as many details of a past design
as the system can provide. Finally, guidelines, the third class of presentation in
Archie-II, are the abstractions behind story lessons; design guidelines generalize
across cases and provide a way of relating parts of cases to one another.

We propose carving cases into chunks of these sorts and allowing users to
browse those chunks in some controlled but natural way as they engage in design.
The three presentation classes — stories, documentation, and guidelines — are
discussed in this section. For each, we will define the class (describing and giving
examples of interesting sub-classes) and justify the inclusion of these classes in
our system. We shall show that documentation helps in organizing the presentation
of the other sorts of chunks so that they can be examined and navigated easily.
In Section 3, we discuss the situations in which it makes sense to present this
information to a user and how individual chunks are indexed.

2.1. STORIES

Complete records of existing complex designs are usually not immediately useful
guides to new design. When architects look to old designs for help on new designs,
they are usually looking for the sorts of help that we have already suggested cases
can provide: ideas for solutions to general problems or specific issues, hints about
pitfalls or opportunities, or reminders of evaluation criteria for parts of their design.
Furthermore, complete records of the design process (including goals, constraints,
options, decisions, evaluations, and revisions) and its outcomes (including inter-
actions, mistakes, and fixes) are exceedingly rare. The fact is, not every aspect
of every building is equally interesting (either absolutely, or in some particular
design context). What is really wanted from experience is not complete exhaustive
records of past designs, but a few good stories that make points relevant to the
current phase of the current design.

The basic idea of a story is that it is a selective presentation about some particular
design case that has some lesson to teach (Schank, 1991) — a lesson more focused
than “this is how to design a building.” Stories pick out related facts from the mass
of information about a design, and marshal the selected facts into a narrative or
argument, highlighting the underlying causation and the critical relationships that
account for some outcome. Stories make good presentation chunks because they are focused enough to aid decision making; we know the kinds of decisions we are trying to help a designer make, and we can pick stories out of our cases that support that kind of decision making.

Stories are an effective means of communicating for several reasons. Instead of simply presenting facts, stories organize facts to make a point, thereby making the facts (as well as the point) more memorable. Stories can illustrate generalizations, giving a better idea of when a rule is applicable (and when it is not), showing how it can be carried out, helping the hearer follow the underlying causality, and exemplifying typical or atypical outcomes. Factual stories can also outdo generalizations as convincing arguments because they claim that the observed outcome truly followed from the stated facts in at least one real situation.

We have identified three different classes of stories that we believe ought to appear in our system: (1) point stories discuss some features of a design, describing how they contribute towards, or undermine, some particular goal or plan for some goals; (2) interaction stories discuss how some features of a design case can be interpreted with respect to several design goals and plans, perhaps advancing some while frustrating others; (3) cluster stories briefly mention the effects of some set of related features in a case without implying any necessary relationship between their various effects. These three classes of story represent different ways of slicing up a design case, suitable for users with different goals. Each is discussed in turn.

2.1.1. Point Stories

Point stories are the simplest sort of story. They deal with the interpretation of design features with respect to a single design goal or plan. The story in figure 2 is an example of a point story. It discusses how the presence of the concession stand in a particular courthouse creates problems for the goal of maintaining a quiet work environment in the courtrooms. As another example, this time of a positive effect, consider the following story:

The Bristol County Courthouse is noteworthy for its successful entrance designs. The building features a separate entrance for juvenile cases to help keep children out of the public eye and away from serious offenders. The entry for accused persons is separate from both the public and the juvenile entrance, and its security is enhanced by use of a sally port. A separate entrance for witnesses was not considered in the design of this building.

In this story, the multiplicity of entrances is analyzed as contributing to the goal that various user groups remain segregated for reasons of security, comfort, and fairness. This story might interest designers considering how to maintain segregation in other circumstances, or those considering the issue of entrance design and wanting to evaluate their decisions' potential effects. In the section on

Analysis due to Anna Zacherl
indexing, we will see how the system can provide access to this story under such conditions. In addition, this story might be of interest as an example of successfully adhering to a particular guideline (discussed in the next subsection), offering the kind of specifics that make it easier to interpret an otherwise abstract rule.

Note that the story does not include all the details of how these entrances were constructed; it only mentions those specifics that support the analysis with respect to the goal of segregation. If, as system builders, we had access to details such as exact entrance dimensions, materials, and construction we would want to make them available in the system, but not as part of this story. Instead, an entrance could be a represented component of the design with such specifications attached; the details of components like entrances are then accessible from within stories that mention them, allowing users who become intrigued by the story to easily see more design details.

2.1.2. Interaction Stories

Analyses describing how design features affect single goals, while often useful idealizations for making a point, rarely capture the full complexity of any particular situation. In many cases, when a designer “makes a mistake” — introduces a design feature that has some negative effect — it turns out that there was some positive reason for introducing that feature. The actual mistake might have been adopting peculiar priorities for goals or failing to realize that a goal was going to be affected by a decision made in the service of another goal. To offer appropriate aid to a user, it is important to understand when such tradeoffs are being made (perhaps without explicit foreknowledge), or when designs exhibit synergies or compound mistakes (Domeshek, 1992).

For example, consider a situation where a designer has chosen black and white signs for a public lobby faced in black and white marble: the problem is that the signs end up blending into the background and visitors to the building fail to receive the guidance they need; the original reason for the design choice was to have the signs fit into the aesthetic scheme of the lobby. A lesson from this situation might be that aesthetic considerations should not be allowed to subvert basic functional performance, and the facts of this case suggest conditions under which this conflict is likely to arise. An interaction story carries the lesson along with details about one way a sign’s function can be subverted (low contrast with background) and suggestion of when to watch out for such interactions (when making decisions about color coordination).

In Archie-II, interaction stories provide access to underlying point stories (and point stories allow the user access to covering interaction stories). The following interaction story from Archie-II’s current corpus discusses several implications of the Bristol Courthouse’s prisoner entrance design, cited in the earlier point story about multiple entrances serving the goal of segregation:

The Bristol County Courthouse has designated entrances for the public, juveniles,
and the accused with the holding area placed immediately inside the accused entrance. The separate accused entrance optimizes security at the entrances and is far enough from the courtrooms to provide good acoustic isolation (and thus a quiet environment for trials). However, security and acoustic isolation of work areas are both compromised during circulation of the accused from the holding area to the courtrooms because the necessarily long path runs adjacent to, and even crosses through, unsecured work areas.

This interaction story discusses the pros and cons of the prisoner entrance design; the point story presented earlier offered an account focused on the success of the design with respect to entrance segregation. Point stories focus on how certain features affect a certain goal, and linked interaction stories make it possible for the user to understand this effect in a larger context. A detailed accounting of the linkages between presentations will be given in Section 3.2.

2.1.3. Cluster Stories

The final class of stories currently included in the design for Archie-II are cluster stories. To a large extent, cluster stories are a browsing convenience. As will be discussed in more detail in Section 3, Archie-II provides several different ways of searching for information, and any of these browsing techniques may initially yield a large set of candidates. The system needs to provide some organized way of viewing the space of candidate presentations to keep the user from feeling overwhelmed. Cluster stories are one way of providing an overview that helps ease the load of browsing such candidate sets.

For example, placing annotations on a blueprint, as in Figure 1, is a way to impose a simple spatial organization on stories. But this is not the only sort of organization we might want, and this simple technique will sometimes break down. When there is a lot to say about one small region of a building or when there is some relationship between several stories located close to one another, a cluster story can stand in for a set of stories, serving as a kind of table of contents organizing the several things there are to say about the region. For example, Archie-II attaches the following cluster story to one particular room:

The court law library is adequate in the Bristol County Courthouse, however, the shelving could be more efficient. The library is also used for showing videos of drivers arrested for Driving Under the Influence of Liquor.

Segments of this text serve as mouseable buttons allowing access to the point stories being referenced: design of an adequate law library; poor design of shelving units; use of the library as a video screening room.

Stories are not limited to clustering in space or on graphics. For example, the following cluster story organizes a set of point stories all related to a particular component of a particular functional system, but one that happens to be spread throughout the building:
In the Bristol County Courthouse, the surveillance cameras throughout the building effectively supplement the work of the court officers responsible for courthouse security.

2.2. GUIDELINES

Stories describe instances of design, and often these can be thought of as illustrations of more abstract rules of design. These rules are rarely firm, so we call them design guidelines. Looked at the other way, guidelines generalize over stories. Like stories, guidelines talk about ways of coping with design issues manifesting in the context of various structural and functional parts of buildings. Unlike stories, however, guidelines do not talk about specific outcomes; they talk mainly about goals, plans, and constraints.

Given this notion of guideline, we can now interpret the “point” of a point story as some commentary on, or interpretation of, a design guideline; a story tells of success or failure in meeting a goal or following a plan, and its concrete details point out pitfalls and opportunities, perhaps speaking to the applicability of the guideline. An interaction story most often reports clashes or synergies that result from the relevance of several different guidelines.

There are several different types of guideline. Some guidelines simply prescribe goals to be achieved, while others map from goals to general mechanisms for achieving those goals or to detailed plans. Guidelines can deal with issues at any grain size, ranging from large strategic decisions to more focused tactical decisions to specification of detailed constraints. Guidelines vary considerably in their force; some describe requirements (perhaps derived from building codes), others describe policies, or simple recommendations. Guidelines may focus on single issues, or may discuss tradeoffs required by goal interactions.

Our first example guideline illustrates how the general issue of safety is associated with a plan calling for a particular type of segregation with specific implication for building layout:2

Judges and jurors should not be subjected to the possibility of prisoner threats, confrontations, or violence. Prisoner circulation should be distinct from that of the court staff.

This guideline maps from some high level goals to a relatively specific suggestion about building layout: keep certain circulation paths distinct. Other guidelines, such as the one below, relate guidelines to each other. This one points out the prevalence of a particular wayfinding problem and thereby simply raises other goals:

Many persons who visit the district courts do so in order to pay fines. It should be clear and obvious where they are to go. As these people enter the building they should be able to find the clerk’s office quickly and easily.

2 The content of this example is taken from a study of courthouses in the state of Michigan. (King, J., Moore, E.O., Johnson, R.E., and Guregian, S.A., 1981).
Here we are told that accessibility of the clerk’s office to fine payers is an issue, but we are simply told to make the office easy to find. Having a space be easy to find is a rather high level goal in itself and we must refer to other guidelines that address this issue. Archie-II contains guidelines telling us that wayfinding to any function of the building by any unfamiliar user of the building can be eased by proper use of signs and by proper location of a function’s space. Part of the more specific guideline about signage runs as follows:

There are many factors to consider when making signs for a building. For instance, disabled users will have special needs that must be considered; problems with vision, stature, and comprehension are particularly important for sign design and placement. Signage both inside and outside such buildings should be accompanied by easily understood symbols. Symbols require neither the ability to read nor knowledge of a particular language; also, when properly designed, symbols can reiterate the message, thus aiding individuals in their task of interpreting sign messages. Signs should be large and should use bold contrasting colors. If signs are to project from the plane of the wall or hang from the ceiling, then they should be mounted with a bottom edge at least 7 feet above the floor. Avoid sharp edges or exposed fasteners on signs. Design lighting that does not cause glare on nearby signs. Place signs to provide maximum visual exposure, particularly along routes of travel.

Finally, after several steps, we have again mapped from high level goals down to relatively specific recommendations for how part of the building should be designed. The point is that the goal/plan decomposition behind guidelines serves as an important means of generalizing across situations and connecting stories with related lessons.

The organization, representation, and presentation of guidelines is an area of Archie-II under active development as this paper is being written. The current approach to organizing this data is to start with a goal/plan hierarchy, rooted at the highest level in an analysis of the functions of particular building spaces over their life cycle from the perspective of particular building user populations. Starting from these high level goals, we can then elaborate a space of guidelines varying in grain size and specificity, in source and in force. We are currently concentrating on fleshing out the part of the guideline space covering accessibility and usability issues, primarily as they arise in courthouses and office buildings for users and staff. We have a long way to go before even this small corner of the larger architectural design guideline space is filled in.

2.3. FACTS AND CONTEXT

So far, with stories and guidelines, we have discussed case materials comprising analyses and evaluations of parts of designs. Archie-II also has to organize and present whatever is available in the way of basic documentation for designs. This documentation includes facts and figures, and it includes graphics of many different kinds; it includes summaries that provide an overview of the entire design, and it
includes detailed specifications for individual parts; it includes the original requirements for the design, evolving requirements, and the eventual design solution that resulted. All this additional information might simply exacerbate the problem of managing efficient browsing; the trick of effective memory organization is to use some of the additional required presentations as organizational tools that actually simplify browsing.

Supporting rapid selective browsing of large case libraries requires the system to provide users with a feel for how much information is available, roughly what its content might be, and where they stand in surveying it. A system should give an overview of the information available — perhaps shaping it into a space, locating the user in the space, and marking off regions that have already been visited; a system should standardize its presentation formats so that users can learn to expect where certain types of information can be found, allowing them to navigate and skim more effectively; a system should, to the extent possible, use familiar presentation formats that users are already adept at comprehending. The presentations used in Archie-II to document basic facts about designs meet most of these goals and thus provide a helpful context for lesson browsing.

2.3.1. Design Graphics

In the domain of architecture, many of the basic facts about a design can be recorded and presented in various graphic forms. It is standard practice for architects to pour over sketches, diagrams, detailed plans, renderings, and photographs of existing or planned buildings; these are presentations we can expect potential users to understand and accept. In fact, use of graphics seems to be standard practice in most design disciplines that deal primarily with the design of physical artifacts, and may be quite common even in the design of processes (where time lines and information flows are often charted).

Graphics are a great example of presentations that serve not only to communicate specific information about a design, but also to help manage the complexity of browsing a design's content; graphics can organize other presentations and help users maintain a sense of what parts of the building they have been studying. Note that even though Archie-II primarily relies on two-dimensional depictions of buildings, graphics are not restricted to representing purely physical context; the parts they depict may also reflect functional systems (such as plumbing, electrical, or HVAC systems) and possibly even design goals (such as zones with particular requirements for privacy, acoustic isolation, or security). Linking graphic displays together in a network that reflects physical contiguity or other logical relationships between parts helps users maintain a sense of location in a space of possible presentations. Locating a story on a graphic (as in Figures 1 and 2) can also serve as a way of identifying the story, helping maintain a sense of what information is available and what has been seen.

Graphics also serve well as context for interpreting other presentations. The
stories brought up by clicking on annotations are not just organized by the graphics, the graphics illustrate the story as well. The story of how noise from the concession stand disrupts court proceedings really only makes sense when you can see the plan of the building showing the concession's location relative to the entrances to the courtrooms.

2.3.2. Design Descriptions

Much design data takes the form of facts and figures, possibly boring but certainly useful in the right circumstances. One of those circumstances is when an architect is just trying to find their way into a browsing system like Archie-II. An architect given a new commission might want to assess the program he or she has been given; scanning existing buildings with similar purposes might allow useful comparison of issues like budgets, aggregate size, and level of finishes. In specific niches, architects are often concerned with idiosyncratic descriptive statistics; for example, when designing speculative office space, important features include percentage of leasable space and maximum lease depth. These are among the kinds of information that ought to be available in design descriptions.

Design descriptions, when serving as a means of finding direction in the browsing process, are probably best presented in groups forming library overviews that summarize some set of available buildings for the user. Such overviews, generated in response to users' initial partial description of their situations, can give an idea of how many relevant cases the system knows about, as well as a feel for the range, variability, and typical values for any selected descriptive dimensions. These overviews ought to serve as the jumping-off point for inspection of more detailed design descriptions, design graphics, and stories.

Detailed design descriptions of the various structural and functional parts of a building are also useful. They can, for instance, provide additional data not included in any story. They can organize data that would otherwise be spread through the text of several stories. While stories are an effective way to organize design information, we cannot expect to anticipate every lesson that might be learned from every building. Sometimes designers will become curious about details beyond those that seemed immediately relevant to a story. As suggested earlier, there ought to be a way to get from a story about entrances to detailed information about those entrances. It also ought to be possible to get from a presentation of the information about an entrance to the stories in which that entrance figures.

3. Indexing

The problem of browsing in Archie-II is not so much a problem of figuring out what sort of chunk to examine or what kind of presentation to look at, as it is a problem of selecting which particular chunk to study. A useful full scale version of the system will have a huge amount of information in it and more chunks than any
one user could browse (especially since a user’s main concern should be designing a building, and not just browsing past cases). The problem of selecting useful items from out of a large memory is a common one in all of CBR; it has been labeled the indexing problem.

The indexing problem is so named because of its similarity to the common problem of finding what you want in a library or in a large book, and because the initial conception of a solution to the problem in CBR is similar to the approach taken to the more familiar problem. To find a topic in a book, you make a guess at a good description of the topic in which you are interested and you look through the index in the back of the book for a listing that matches; if the contents of the book are well indexed and if you have guessed a good probe, then the index will tell you where to find relevant material. In CBR, the approach to finding a case (or a chunk of case) in a case library is similar, though the nomenclature is different: in CBR, the equivalent of a book’s index is called an indexing system, while the description you use as a probe and the individual entries in the indexing system are called indexes. Nonetheless, the problem of access to appropriate chunks is the same, and the dependence on arriving at good descriptions of those chunks is similar.

Archie-II uses two different styles of indexes. The kind we have been talking about so far are called descriptive indexes. These indexes are intended to support reasonably flexible forms of search through the corpus. The issue raised by such indexes is how to design a simple but comprehensive language for describing the conditions under which chunks are worth retrieving. The other kind of indexes are relationship indexes. Relationship indexes are best thought of as being more like the kind of pointers authors embed directly in their texts — for example, “see Section 3.2 below for more discussion of relationship indexes.” Relationship indexes only make sense when it is possible to make a limited set of reliable predictions about what sorts of information a user should want to see next when in a particular context.

3.1. DESCRIPTIONS

Stories are the chunks we would most like to index by descriptions; they most directly achieve the purposes we defined for memory chunks: proposing solutions, identifying pitfalls and opportunities, and suggesting evaluation criteria. To settle on a way of describing stories, we need to figure out what stories can be about. The major thing that distinguishes a story from any other chunk of a design is that it selects and organizes design features in order to account for an outcome with respect to some instance of a design issue. That means we have to worry about describing design issues and the contexts in which they give rise to more specific goals. A story can be about safety in the courtroom (a design issue and a structural component). A story can be about the efficiency of the air conditioning system (a design issue and a piece of a functional system). A story can be about privacy
in the holding cells as affected by the surveillance cameras (a design issue in the context of both a structural component and a functional system). To complete the mapping from design issue to design goal, we often need to pay attention to who cares about the issue, and when it matters. Safety means different things to a judge and a prisoner; it means different things to people using the building than to those building the building.

The need to map from design issue to design goals generates our sketch of Archie-II's descriptive indexes. The indexes are composed of descriptors specifying a design issue and some of the other four components, listed here (including at least a structural or functional building part):

- a design issue;
- a structural component;
- a functional system;
- a stakeholder perspective;
- a phase in the lifecycle.

Design issues are the highest level concerns that apply in particular and sometimes idiosyncratic ways to every building and to many parts of buildings. Examples issues include efficiency, accessibility, usability, safety, comfort, privacy, symbolism, and aesthetics. The structural component descriptor may refer to either a generic class of structure (floor, wing, room), or a specific structural component (the main floor, the north wing, the staff lounge). The functional system descriptor may refer to either a generic system (plumbing, electrical, HVAC), or a specific component from such a system (a pipe, valve, switch, duct, blower, thermostat). These are the sorts of building parts about which architects must make design decisions, so these are the kinds of parts about which the system must be able to tell stories. Stakeholders are the various classes of individuals and institutions that have concerns about the building. Stakeholders include the designer, builder, owner, community, residents, users, mission-staff, and support-staff. The lifecycle phases during which these stakeholders are likely to have particular goals for the building include construction, use, maintenance, and renovation.

We can use this system to describe some of the stories used earlier as examples. For example, the original point story about the value of several entrances can be described in the following two ways.

<table>
<thead>
<tr>
<th>Design Issue</th>
<th>Privacy</th>
<th>Safety</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structural Component</td>
<td>Juvenile Entrance</td>
<td>Prisoner Entrance</td>
</tr>
<tr>
<td>Functional System</td>
<td>Juveniles</td>
<td>Community</td>
</tr>
<tr>
<td>Stakeholder</td>
<td>Use</td>
<td>Use</td>
</tr>
<tr>
<td>Lifecycle Phase</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As a more complex example, consider the interaction story built around the courthouse entrance design; that story might be described as a tradeoff with design features that successfully ensure security at the prisoner entrance undermining
security within other working parts of the building. To build this description we need to introduce one additional descriptor to the format. Since an interaction story is a discussion of how several issues were differentially affected by some design features, descriptions of interaction stories will need to be able to describe more than one issue and its outcome. In this example, the two columns are to be read not as two different descriptions, but as a tradeoff description, the first column a positive outcome, the second negative. We expect to limit such descriptions to pairwise tradeoffs each mentioning only two goals and outcomes.

<table>
<thead>
<tr>
<th>Design Issue</th>
<th>Structural Component</th>
<th>Functional System</th>
<th>Safety</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholder</td>
<td>Prisoner Entrance</td>
<td>Community</td>
<td>Safety</td>
</tr>
<tr>
<td>Lifecycle Phase</td>
<td>Use</td>
<td>Use</td>
<td>Prisoner Path</td>
</tr>
<tr>
<td>Outcome</td>
<td>Positive</td>
<td>Negative</td>
<td></td>
</tr>
</tbody>
</table>

The information described here as potentially belonging in a story description is intended to support three major uses of stories: proposing solutions, identifying pitfalls and opportunities, and suggesting evaluation criteria. Given a goal, it is possible to search for descriptions that have that goal and a positive outcome to find stories that tell of solutions for the goal. Given information about a part of a design, it is possible to search for descriptions mentioning that part and having either positive or negative outcomes to find stories that identify pitfalls and opportunities. Given both a goal and a part of a design, it is possible to search for descriptions with either positive or negative outcomes to gather stories that between them should suggest additional features that determine difference in outcome and hence make for good evaluation criteria.

3.2. RELATIONSHIPS

Relationship indexes are simpler than descriptive indexes. They are essentially hard links between presentations intended to direct the user’s attention to related material. In any situation, Archie-II can make some reasonable predictions about what chunks would be worth seeing next and can make those recommendations visible to the user as browsing options.

The number of relationship types is small and fixed. In the original discussion of the various chunks, mention was made of the sorts of transitions between presentations that we intended to support. Figure 3 gathers all of that information in one place. To summarize briefly:

- point stories link to interaction stories in which they participate, and interaction stories link to their underlying point stories;
- point and interaction stories link to guidelines whose application or failure they illustrate, and guidelines in turn link back to the stories over which they generalize;
cluster stories can link to either point or interaction stories which link back to their clusters;

- library overviews link to design graphics, design descriptions, and stories, all of which link back to overviews, and to one another.

One way to think of these links is as part of the system's interface, but they can also be thought of part of the system's memory network. The catch, of course, is that this memory is primarily intended for perusal by a human user, rather than for autonomous search and manipulation by the system itself. The point is that the content described above and the organization pictured here are useful in supporting the design process.

4. Project Status

Archie-II currently exists as a set of twenty analyzed stories and their accompanying guidelines, a preliminary vocabulary for describing those stories, and an interface prototype developed in Supercard™ on an Apple Macintosh™. The system's stories are drawn from a set of Post-Occupancy Evaluation reports prepared by architectural consulting firms as part of procurement review processes of various government agencies. The guidelines come from existing compilations of guidelines, codes, and expert analysis.

The current interface supports presentation of the several types of case chunks discussed above. While it does not currently implement all of the different presentation modes, it gives a good feel for the sort of interaction we have in mind. We are in the process of using this interface prototype to solicit feedback on the design from human/computer interaction and visualization specialists, and from architects.
who constitute the system's potential user community. We are asking them to evaluate proposed functionality, the organization of that functionality on screen, and the vocabulary available to users for indexed searches. So far we have received much interest and encouragement from architects who have seen the system.

By the time this paper appears, we will have been through several iterations of the interface design and should have a system with several dozen stories organized in its memory. Current areas of active development include the accumulation of more stories and guidelines, and the further elaboration of indexing formats and vocabulary. These activities are closely interrelated, since in the case-based paradigm, it is the existence of data to be indexed that drives the development of indexing formalisms.

Conceptual design is both intellectually fascinating and economically important. We believe that by taking a case-based aiding approach to conceptual design, we will be able to learn something about the process of design, and also have a positive effect on the product of design. It is this mixture of scientific and technological thrusts, each with short term and long term payoffs, that make us optimistic about the future of Archie-II and systems like it.

Acknowledgements

Many people have worked on the development and conception of the systems described in this paper. Craig Zimring, our colleague in architecture, and Richard Billington, our research programmer, have been constant collaborators throughout. The Archie-II team also includes Joshua Bleier, Anna Zacher!, and Kadayam Vijaya. An earlier version of the system, Archie, was developed with the help of Ashok Goel, Michael Pearce, and Lucas Sentosa; lessons learned from that effort have substantially influenced our current direction. This work has been supported in part by the Defense Advanced Research Projects Agency, monitored by ONR under contract N00014-91-J-4092. All views expressed are those of the authors.

References


Case-Based Design Support:
A Case Study in Architectural Design

Michael Pearce
Ashok K. Goel
Janet L. Kolodner
Craig Zimring
Lucas Sentosa
Richard Billington

July 6, 1992

Pearce is with Galaxy Scientific, Atlanta; Goel, Kolodner, and Billington are with the College of Computing, Georgia Institute of Technology; and Zimring and Sentosa are with the College of Architecture, Georgia Institute of Technology. Contact: Ashok Goel, College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, Georgia 30332. email: goel@cc.gatech.edu. phone: (404) 853-9371.

This paper is to appear in IEEE Expert, Special Track on Case-Based Reasoning
Case-Based Design Support: 
A Case Study in Architectural Design

Michael Pearce, Ashok K. Goel, Janet L. Kolodner, 
Craig Zimring, Lucas Sentosa and Richard Billington

1 Introduction

Designing complex artifacts generally involves making a series of decisions. Designing an office building, for example, may involve decisions about instantiating abstract design concepts and skeletal design plans, selecting specific systems and components, retracting earlier decisions, relaxing design constraints, accepting a design solution, etc. Depending on the number of possible choices at a decision point, the knowledge available to decide on a specific choice, and the interactions among different decisions, making such decisions can range from easy to very hard. In designing an office building, for example, the number of possible spatial configurations for the building may be very large, and there may be no simple correspondence between the design constraints and the various configurations, making the selection of a specific configuration for the building very difficult. If the architect does not have appropriate knowledge that can guide his decision making, this situation can easily result in poor decisions and thus in poor designs. Supporting architects and other designers in making good design decisions therefore is a real challenge for research on intelligent computer-aided design.

Informal observations of architectural practice, discussions with practitioners and teachers of architecture, and examinations of architectural literature, all indicate that architects often base their design decisions on past design cases. Suggestions made by past cases help the architect with such tasks as elaborating design problems, proposing design solutions, critiquing proposed designs, modifying proposed designs, explaining and justifying
specific design choices, and refining design solutions.

Yet, despite the apparent abundance and extensive use of past designs in decision making, individual architects do not always have ready access to the past designs appropriate to a given design task. Architectural design cases are not organized in any general library, but are scattered throughout file cabinets, magazines, books, and the memories of other designers, making access difficult. Some large architectural firms do have small libraries of designs created by the firm. However these design libraries typically are indexed by a small number of mnemonic features that are not always indicative of the relevance of a past design to a given design problem, making it difficult to find the right case, or the right piece of an old case, to guide decision making.

Several years back, these observations led us to hypothesize the creation of a large computer-based library of architectural design cases [Goel et al. 1991]. This computer-based library, we envisioned, would provide architects with a non-volatile external memory of design cases. In order to provide some focus to our work on creating this case library, we made three initial design decisions. First, we decided on a specific division of responsibility between the computer system and the user in which the system supports a range of common decision-making tasks in design but leaves all decisions to the user. Second, we selected the design domain of office buildings for constructing the case library. We chose the domain of office buildings because we had access to experts in this domain and also because as naive users we have some experience with office buildings. Third, we decided to build a case library to support conceptual (or preliminary) design as opposed to detailed design. We chose to focus on conceptual design because in general this is the more difficult and innovative phase of the design process. In addition, since conceptual design is done early in the design process, decisions made at this stage have a major impact on the rest of the process. Within the context of conceptual design, we decided to support two decision-making subtasks of design: proposing solutions to a design problem, and critiquing a proposed solution.
Our work has led to the development of a small computer-based library system that we call ARCHIE. When we started working on ARCHIE, we thought that we would need to address three main issues: how to represent architectural design cases, how to index them, and how to retrieve them. As a result, much of our early work focused on developing a vocabulary for representing architectural design cases and organizing them in a library, and on schemes for retrieving cases relevant to a given design task. In the process of building and using ARCHIE, we encountered additional issues whose importance we did not fully appreciate in the beginning. This paper describes how we built ARCHIE, what it does and how, what issues and problems we encountered in developing it, and what lessons we learned from it. The lessons learned from ARCHIE suggest new research directions that we are presently exploring in two parallel projects: ARCHIE2 [Domeshek & Kolodner 1992] and ArchieTutor.

2 Illustrative Examples from ARCHIE

ARCHIE helps architects in the high-level task of conceptual design as opposed to low-level tasks such as drawing and drafting, numerical calculations, and constraint propagation. The operation of ARCHIE is illustrated by the examples described in the following two subsections. The first example shows how ARCHIE helps an architect in creating a high-level conceptual design for an office building. In this mode of operation, the system provides the architect access to designs of office buildings created by other architects in the past, and points to various factors that need to be considered in solving a given design problem. The second example shows how ARCHIE helps the architect in critiquing a proposed conceptual design for an office building. In this mode of operation, the system provides the architect with access to the perspectives of various domain experts, and enables the architect to anticipate and accommodate their views.
2.1 Design Proposal

A session with ARCHIE begins with a client describing a set of requirements that a desired office building should meet. In one example from ARCHIE, the client gives the architect the goal of designing an office space for a small accounting firm. Since customers frequently visit the company offices, the client feels that the image the office space projects to customers is very important. Given the nature of the work to be done in the office space, facilities for face-to-face contact and group interaction among the workers also are important. Also, the company is dynamic and growing, and so the final design should be flexible enough to allow modifications. In addition, since the company maintains paper files on its clients, it is important that the offices have a good filing system. The client also specifies the constraints that the building should meet. For example, the company requires about 12000 square feet of space, and has budgeted $300,000 for the partitions and furniture system.

ARCHIE provides a vocabulary for specifying these design goals and constraints. The architect uses this vocabulary to represent the design goals and constraints as illustrated in Figure 1. Each of the boxes in the figure represents a feature in the probe, where the text on the left side is the name of the feature, the text on the right side is the value of the feature, and the italicized text to the right of the each box describes the type of value that feature can take. The first group of features in the figure prefixed by "O-GOAL" describe the type of organization that will be using the building. The "C-GOAL" and "F-GOAL" features specify the constraints on the core (the permanent structure of the building) and the furniture of the building, respectively.

ARCHIE uses the specified goals and constraints for the desired design to probe its
case memory, and retrieves cases that are most similar to the probe. Figure 2 illustrates a part of one of the retrieved cases. The first column in the figure contains the features describing the design goals and the constraints on the design. The second column specifies features pertaining to the design plan for the office building in terms of its physical core, partitions, and furniture. The third column contains features describing the outcome of the design of the building, i.e., an evaluation of the design from different points of view such as lighting, group interaction, and privacy.

As we mentioned above, the design goals and outcomes prefixed by a "O" pertain to the client organization as a whole. These features describe the group of people that use the building, and their interactions with that building. The design goals, plans, and outcomes prefixed by a "C" describe the core of the building. The core features describe the permanent aspects of the building. For example, the total area of the design described in Figure 2 is about 14500 square feet and is likely to remain the same over time. Features prefixed by a "P" specify the partitions in the building. These features deal with the non-permanent interior structures that divide the space inside the building. For example, one aspect of the design plan shown in Figure 2 is that the manager's office has glass walls; this aspect of the building design may well change over time. Features prefixed by a "F" represent the furniture in the building. These features pertain to the furniture in the building. For example, one outcome of the design illustrated in Figure 2 is that the furniture color was found to be pleasing.

The architect can browse through the retrieved cases and see the ways in which other architects solved similar design problems. She may decide to combine design solutions from several cases to create a high-level qualitative design that achieves the goals specified by the client. For example, she may notice that the case shown in Figure 2 uses an open office plan to achieve the goals of flexibility, face-to-face contact, and group interaction. An open plan in architecture implies low, movable partitions to separate office spaces rather than fixed
ceiling-to-floor walls. Since face-to-face contact and group interaction among the workers are important design goals in the current problem, the architect may adopt the open office plan of the previous case. The architect now copies features from interesting cases to produce a partially specified conceptual design for the new office building. In this way ARCHIE helps the architect to brainstorm: it enables the architect to explore previous cases for ideas and inspirations for solving a problem.

2.2 Design Critiquing

Now that the architect has a partially specified conceptual design for the office space, she may want to critique the design to determine if there are any potential problems with it. This involves more focused reasoning: the architect is interested in accessing detailed information within a narrow context. To illustrate, consider a second example from ARCHIE in which the architect is concerned with the quality of lighting in the proposed design for an office building. ARCHIE contains several domain models that represent the perspectives of different domain experts. Critiquing is done by selecting one of these views of ARCHIE, where each view contains a single qualitative model that evaluates a single aspect of the design. Since, in this example, the architect is concerned with lighting quality, she selects the lighting view.

ARCHIE organizes the cases in its memory into clusters of cases that resulted in specific types of outcomes (more on this later). Since the architect selected the lighting view, ARCHIE compares the cases in the lighting cluster with the proposed design and retrieves the cases similar to it. Figure 3 illustrates a part of such a retrieved case. It shows the features (design goals, critiques of the design plan, lessons learned, etc) that pertain to lighting. The morals and lessons are expressed in the form of free text and act as annotations to the cases. Figure 3 shows the annotations in bold face. The annotations labeled Goal Comments in the figure refer to the interactions among the design goals; Plan Comments
specify individual design decisions and their relations with the design goals; and Outcome Comments explain the outcomes. The Case Comments provide general background on the design of the office building. The O-Outcome Moral specifies the lessons learned from the design case. The lessons from the design case in Figure 3 are enclosed in a double box.

Note that the C-OUTCOME: Lighting Satisfaction-Artificial feature enclosed in a double box on the right side of Figure 3 shows that the artificial lighting in the retrieved design caused glare problems. ARCHIE retrieved this design because of its similarity to the proposed design, and this helps to warn the architect that the proposed design may also have glare problems. The O-OUTCOME: Moral enclosed in a double box on top of the figure indicates how glare problems can be avoided, viz. by using diffusers over bulbs. This tells the architect of a way to remedy the proposed design to avoid glare problems. Once the architect has fixed the design, she may similarly critique the design from other perspectives. This process of retrieving, critiquing, and adapting continues until the architect is satisfied with the proposed design.

ARCHIE thus supports architects in both proposing design solutions and critiquing them. Further, it acts as a shared external memory that supports two kinds of design cooperation [Goel, Kolodner, Pearce and Ziring 1991]. First, by including enough knowledge about the design goals, plans, outcomes and lessons in past cases, it provides the designer access to the work of previous architects. Second, by providing access to the perspectives of various domain experts via the domain models, ARCHIE enables the architect to anticipate and accommodate their views on the evolving design.

3 System Implementation

ARCHIE was constructed using the CBRShell from Cognitive Systems, a case-based knowledge engineering shell with several tools for developing applications using case-based technology [Cognitive Systems 1990]. The tool allows the programmer to design a case rep-
presentation. Enter the case information into the shell, set up a retrieval mechanism for the system, and design a presentation of the case features. The user may then enter information about the current design problem, specify a retrieval method for the case-base, and retrieve cases from memory. The tool has several nice features, including a graphical function editor for dynamic feature computation, direct manipulation editor for case display design, and a retrieval explanation editor. All figures in this paper are from ARCHIE screens that were produced using the CBRShell. They represent part of what the user would see when interacting with the system.

3.1 Sources of Cases

ARCHIE contains about twenty design cases. These cases came from three main sources. The first source is our own personal experiences. These cases are based on our naive analysis of office buildings in which we (one or more of the authors) have worked. The second source of cases is professional architectural journals. Typical journal articles are usually fairly short and deal with some specific aspect of the building such as aesthetics, furniture systems, lighting systems, etc. The third source of cases in ARCHIE is Post Occupancy Evaluations (POEs) done by professional architects and researchers. POEs are usually done when a building exhibits some major problem, and attempt to identify the sources of the problem, and suggest several ways (based on cost and effort) of fixing them. The quality and completeness of information in ARCHIE's cases thus varies widely.

4 Knowledge Organization

ARCHIE contains three types of knowledge: design cases, primitive concepts, and domain models. Design cases contain ARCHIE's knowledge of the designs of specific office buildings. Primitive concepts refer to the primitive objects, relations, and parameters of office buildings, and form part of the language for representing and indexing cases. The domain models
capture the causal relationships among the concepts of the cases.

4.1 Cases

A case in ARCHIE is represented as a flat, static frame with over 150 possible features. The features are of several different types: concepts, text, integers, real numbers, and functions act as fillers of frame slots. The concept slots (discussed below) are the most important for case retrieval because they allow for the representation of classes of objects and their partial orderings. Most of the features in the case representations are concepts, with some numbers and some free text.

The features originate from three sources. Some features came from our analysis of several office buildings. We built a set of features which can describe office buildings with sufficient fidelity so that one office building can be differentiated from another. These include features that describe the people that use a building, the rooms and equipment inside that building, and the things the users liked and disliked about the buildings. Some features came from the domain models that described the various outcomes of the buildings. For example, the color of the walls, floor, and furniture all effect the amount of glare in an office, so these features were added to the representation language. Finally, some features came from our analysis of usability issues. Trying to understand a case by reading the concept features in its description is often difficult because of the large number of features. Since the user should be able to understand the points that a case is trying to make without having to understand our representation of the cases, we added textual features to give a more natural interface to the case representations. We found that the number of descriptive features in ARCHIE grew rapidly as we analyzed the first few architectural design cases but was fairly stable after we had represented about 10 cases.

A design case in ARCHIE contains several types of information [Goel et al. 1991]:

- Design goals and constraints. This component describes the organization that uses the
building and the constraints on the design. As shown in the first column of Figure 2, an example of a goal in designing office buildings is the requirement for a large amount of group interaction, and an example of a constraint is that the partition height should be above eye level.

- **Design plan.** The plan component of the case captures knowledge about the plan designed to satisfy the client’s needs. A design plan specifies building components (walls, doors, furniture, etc.) and their configuration. The second column in Figure 2 illustrates the design plan of a case.

- **Outcome of the design solution.** An outcome is an evaluation of the plan. An outcome describes how well the plan satisfied the design goals and constraints (explicit and implicit) from a specific point of view. For example, a design outcome could be that auditory privacy was low, as indicated in third column of Figure 2.

- **Lessons and morals learned from the design outcomes.** These are free-text annotations that describe the lessons that the case holds for the architect. They provide an easy-to-understand summary of the outcomes of the case, and indicate in which situations the case will be useful. The case shown in Figure 3, for example, contains the lesson that “offices with fluorescent lighting systems should have diffusers over bulbs to avoid glare”.

### 4.2 Case Retrieval

The CBRShell provides ARCHIE two mechanisms for retrieving cases: nearest-neighbor matching and model-based clustering. These two retrieval mechanisms are used in the brainstorming and focused problem solving modes of operation respectively, as described in the examples of Section 2.
4.2.1 Primitive Concepts and Nearest-Neighbor Matching

Case retrieval by nearest-neighbor matching is based on the hierarchical organization of the concept features that form ARCHIE’s language for specifying goals, plans and outcomes of design cases. Some of ARCHIE’s concepts pertain to domain objects such as lighting devices; others are more abstract, for example, lighting quality. A small subset of these concepts is shown in Figure 4. Concept nodes with boxes on the right side in the figure have subnodes that can be expanded into the “values” that the concept can take. Thus ARCHIE knows that lighting devices can be of different types: fluorescent, halogen, and incandescent. The CBRShell allows for a partial ordering of concepts, as indicated by the “<” relation between some concepts in the figure.

ARCHIE uses these concepts in judging the similarity between a user-specified probe and a stored case. To illustrate, let us return the example described in Section 2. where architect had used the vocabulary of ARCHIE’s concepts to prepare the probe shown in Figure 1. ARCHIE accepts this probe and calculates a similarity value for each case in its memory. The similarity value of a case to the probe is based on the closeness of each of the concept values in the case to the specified concept values in the probe and on the importance assigned to the concept. The system retrieves cases whose similarity value is above some predetermined threshold. Figure 2 illustrates a case that was retrieved by the probe shown in Figure 1. ARCHIE uses nearest-neighbor matching whenever an architect specifies the goals and constraints of the current problem, and wants to retrieve designs of buildings that satisfy goals and constraints similar to the current problem.

4.2.2 Domain Models and Model-Based Clustering
ARCHIE’s second retrieval method uses simple domain models for clustering the cases in memory. These causal models represent ARCHIE’s domain knowledge about the design of office buildings. Figure 5 shows ARCHIE’s causal model of lighting. This model captures qualitative knowledge of how the features in the design of an office affect the lighting quality. According to this model, satisfaction with the lighting is dependent on the quality of light and on the glare intensity (the nodes with rounded corners). The glare intensity in turn depends on the shade of the wall color, the shade of the floor color, etc.; these nodes on the left of the figure are features of the cases. The positive (and negative) signs on the arrows in the figure indicate that the value of the feature being pointed to increases (or decreases) with an increase in the value of the feature that the arrow points from. For example, as the shade of the wall color “decreases” from darker to lighter shades, the glare intensity increases, which leads to a decrease in the overall satisfaction with the lighting. The lower half of the model captures knowledge about the factors that lead to glare in an office room.

ARCHIE uses these causal models when an architect wants to critique a design from some perspective. To illustrate, let us again return to the example in Section 2, where an architect wanted to critique a proposed design from the viewpoint of lighting quality. She selects the lighting model shown in Figure 5. ARCHIE uses this model to cluster cases by the features that affect the lighting outcome (the features on the left side of the lighting model), and the resulting cluster is shown in Figure 6. This figure shows the classification tree for the cases in ARCHIE’s memory that was derived from the lighting model. The rectangular boxes in the figure specify the partitioning nodes in the classification tree, and the text in these boxes gives the name of the corresponding feature in the lighting model. The boxes with rounded corners represent the actual case clusters, and the text in the nodes describes
the outcomes of the cases in each of the clusters. For example, the rounded box with the text “1 Glare CASE” and its placement in the classification tree indicates ARCHIE’s memory contains only 1 case with a low lighting quality and a high glare intensity.

ARCHIE uses the lighting model to predict the outcome of the lighting quality in the proposed design based on the design features. For example, ARCHIE may predict that the proposed design will result in low lighting quality and high glare intensity. Since this outcome matches the “1 Glare CASE” in the classification tree shown in Figure 6, ARCHIE retrieves this case (this case is illustrated in Figure 3) and displays it to the architect. In this way, ARCHIE’s models help to retrieve cases that can help an architect in critiquing a proposed design.

5 Lessons Learned

Our initial work on ARCHIE was focused on the issues of representation, indexing and retrieval of design cases. In the process of building and using ARCHIE however we encountered several additional issues which hold practical lessons for building large-scale case-based systems for supporting real-world decision tasks.

Lesson1: Incompleteness of Real-World Cases. When we started working on ARCHIE, we knew that the domain of architectural design is characterized by an abundance of design cases. In the process of building ARCHIE, however, we found that despite the abundance of design cases, it is not easy to find well-documented cases of the design of office buildings. While drawings and pictures of office buildings are readily available, the available analysis of these cases typically is incomplete. Except for some atypical office buildings, the architect’s justifications for the design decisions and the outcomes of the design generally are not known. Indeed, in many cases, even the specification of the design goals and constraints is not clear. This severely limits the potential usefulness of old design cases in making new design decisions. It is also one of the reasons why ARCHIE contains only about 20 cases.
This knowledge acquisition problem raises the issue of how to acquire well-documented cases and how to use incomplete cases to support decision making. Since some of the information that would be useful to have is just so difficult to obtain, the best practical solution might be to settle for a large number of semi-documented cases rather than attempting to fully document a few cases. Such an approach will require robust reasoning procedures that can process incomplete and possibly inconsistent case descriptions. It may, for example, require procedures to combine information in multiple cases in a manner that detects inconsistencies.

**Lesson 2: Large Size of Real-World Cases.** ARCHIE provides a language for representing an office-building design case. In addition, it shows how this language also provides a vocabulary for indexing design cases and specifying design problems. As indicated by Figures 1, 2 and 3, the vocabulary of this language is quite large (it contains more than 150 conceptual primitives). This large vocabulary is needed to capture enough information about a case so that the case can be useful in a number of different situations. A well-documented case may contain a huge amount of information that is potentially useful to architects: information about the clients, design goals and constraints, the designers, the design history, the design plan, post-occupancy feedback from the users, analysis of their feedback, etc. The large size of cases is another reason why ARCHIE contains only about 20 cases: much effort is required to gather enough information about a design to make the case useful in a variety of situations.

The sheer size of real-world cases presents a variety of problems for building a case-based decision-support system, ranging from representing the content of cases, to indexing cases in memory, to presenting cases to an architect. For example, it results in a large gap between the grain size of the available cases and the grain size of decisions that the architect needs to make. Since ARCHIE presently contains only large grain cases, it is not easy to extract fine-grained information from it.
In ARCHIE2 [Domeshek and Kolodner 1992], we are presently investigating the prospect of organizing large cases into case-subcase hierarchies [Barletta and Mark 1988] [Redmond 1990] as a potential solution to the problem posed by the large size of real-world cases. This solution however raises several complex issues of its own. One important issue pertains to the dimensions in which a case can and should be decomposed. The JULIA system, which designs menus for meals, for example, decomposes meals structurally [Hinrichs 1991]. The KRITIK system, which designs electrical and mechanical devices, decomposes devices both functionally and structurally [Goel 1991]. A preliminary analysis of the design of office spaces suggests they too can be decomposed both structurally and functionally. For example, an office building may be decomposed structurally into different floors, and the different floors can be further decomposed into different offices, corridors, etc. The same office building may also be decomposed functionally into different types of offices and other spaces, and how they are used.

The organization of cases into case-subcase hierarchies raises yet another issue, namely, how to present information about the internal organization of a case so that the user of the system can easily navigate the subcases within the case. The format for presenting this information must enable navigation both within a case and across several cases.

**Lesson3: Access to Multiple Types of Knowledge.** Since the initial goal of this project was to investigate the role of case-based reasoning in the design process, ARCHIE provides an architect with an external memory of design cases only. The domain models in ARCHIE therefore are not accessible to architects. However as this project progressed, we realized that access to the domain models too may be useful to architects especially because they often appear to use multiple types of knowledge in creating designs of office buildings. We believe that access to domain models in addition to design cases may be especially useful for architectural students because they make explicit interactions among objects and processes in the domain.
In ArchieTutor, we are investigating the prospect of representing design cases, domain models, and design rules, and of allowing the architect access to the models and rules in addition to the design cases. This too raises several new issues. One important issue is that of cross-indexing between different types of knowledge, so that the user can navigate from any one type of knowledge to any other type, for example, from a design case to a domain model that explains some portion of the case, to a design rule that is an abstraction of the model, to another case that illustrates the rule. KRITIK [Goel 1991] provides a scheme for cross-indexing design cases and causal models. AskJef [Barber et al 1991], which helps software engineers in designing human-computer interfaces, provides a scheme for cross-indexing design cases and design rules.

Another important issue in this context is that of providing direct access to design rules and domain models in addition to the design cases, so that, given a design decision that the architect is facing, the system may present some combination of cases, rules, and models relevant to the decision. This direct access to different types of knowledge requires (i) a taxonomy of decisions and decision contexts that the prospective user may face, and (ii) an indexing scheme that maps classes of decisions and their contexts into the different types of knowledge available in memory.

**Lesson 4: Presentation of Relevant Information.** Finally, as illustrated in Figures 2 and 3, ARCHIE’s cases contain several types of information: descriptions of the organization that uses the building, problems that the building has had, and the lessons that the case teaches, etc. ARCHIE uses a set of static “forms” to present all this information to the architect. Each form corresponds to a view of the system (e.g., the lighting model), and specifies a subset of the case features and the arrangement for displaying them. The system gives the architect a choice of such forms, and the architect must select the form relevant to her particular task.

Since ARCHIE uses static forms for presenting information, neither the architect nor
the system can dynamically restructure the interface to make important information more salient. However, ARCHIE's cases can be used both for design proposal and for design critiquing, and different chunks of information are more or less important in different phases of the design process. ARCHIE nevertheless uses the same static forms for both design proposal and critiquing.

Also, since much of our early work was focussed on representation issues, the interface was designed more for knowledge engineering than for use by architects. It soon became obvious that the interface was too painful to use by anyone other than a knowledge engineer. Even if ARCHIE had lots of well documented cases, it would not be usable unless it presents information in a format that enables the architect to quickly and easily understand the information. Our current research is exploring such usability issues and especially the issue of how to best present large amounts of knowledge to a user [Domeshek & Kolodner 1992].

6 Conclusions

Kolodner [1991] has proposed building large case libraries for supporting and improving human decision making in complex tasks such as design. ARCHIE represents an initial experiment in substantiating and evaluating this proposal for a practical task. ARCHIE accomplished many of the goals that we had in mind when we started the project. It provides a vocabulary for representing and indexing architectural design cases in a computer-based library. This vocabulary has stabilized over the course of construction of the system and appears to be expressive enough to represent and index additional cases. ARCHIE also provides mechanisms for retrieving cases relevant to the tasks of proposing conceptual design solutions to new problems and critiquing proposed solutions. In addition, ARCHIE provides a shared external memory that supports two kinds of design cooperation: it provides the designer access to the experiences of previous architects, and to the viewpoints and knowledge of different domain experts.
It is also clear, however, that the current version of ARCHIE is only partially success-
ful in accomplishing its main goal of supporting architectural design. Indeed, some of the
most important contributions of the ARCHIE experiment are the lessons we learned from its
limitations. We learned that real-world cases typically are poorly documented. This limits
their usefulness for case-based decision support, and raises the issue of combining informa-
tion from multiple cases. Further, well-documented cases generally are very large in size.
This raises the issue of decomposing and organizing cases into subcases. In addition, we
learned that the issues of human-computer interfaces and presentation of information are of
critical importance in building an usable case-based decision support system.

Acknowledgments

The group working on the original ARCHIE system included Sambasiva Bhatta, Thomas Hinrichs,
Ali Malkawi and Daniel Willham at different stages of the project. In addition, this paper has
benefited from subsequent work on the ARCHIE2, ArchieTutor and AskJef projects. The ARCHIE2
team includes Richard Billington, Joshua Bleier, Eric Domeshek, Janet Kolodner, Anna Zacherl
and Craig Zimring. The ArchieTutor team includes Ashok Goel, Kim Liu, Ali Malkawi and Michael
Pearce. The AskJef group included John Barber, Sambasiva Bhatta, Ashok Goel, Mark Jacobson,
Michael Pearce. Louise Penberthy, Murali Shankar, Robert Simpson and Eleni Stroulia. Work on
the ARCHIE project has been supported by a grant from the Defense Advanced Projects Research
Agency, monitored by AFOSR under contract F49620-88-C-0058.

References

Shankar, R. Simpson, E. Stroulia. AskJef: Integration of Case-Based and Multimedia Technolo-

Workshop on Case-Based Reasoning, 1988, pp. 12-17.

[Cognitive Systems 1990] Cognitive Systems, Inc. Case-Based Retrieval Shell; Users Manual, Jan-
uary 1990.

[Domeshek and Kolodner 1992] E. Domeshek and J. Kolodner. A Case-Based Design Aid for Ar-
chitecture. In Proceedings of the Second International Conference on AI in Design, Pittsburgh,


20
Figure 1: A Probe into ARCHIE's Case Memory
### Goal

<table>
<thead>
<tr>
<th>Organization</th>
<th>Core</th>
<th>Partition</th>
<th>Furniture</th>
</tr>
</thead>
<tbody>
<tr>
<td>O-GOAL: Organization Type</td>
<td>C-GOAL: Core Type</td>
<td>P-GOAL: Partition System</td>
<td>F-GOAL: Furniture Budget</td>
</tr>
<tr>
<td>Corporate Back Office</td>
<td>Open</td>
<td>Unknown</td>
<td>20000</td>
</tr>
<tr>
<td>O-GOAL: Group Interaction</td>
<td>C-GOAL: Core Area</td>
<td>P-GOAL: Partition Height</td>
<td>Unknown</td>
</tr>
<tr>
<td>Very Important</td>
<td>15000</td>
<td>Above Eye Level</td>
<td>Unknown</td>
</tr>
<tr>
<td>O-GOAL: Image Importance</td>
<td>C-GOAL: Total Area</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Important</td>
<td>14500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>O-GOAL: Information Sources</td>
<td>C-GOAL: Ambient Lighting Type</td>
<td>P-GOAL: Number of Conference</td>
<td>760</td>
</tr>
<tr>
<td>Face to Face</td>
<td>Flourescent</td>
<td>Workgroup Division</td>
<td>1</td>
</tr>
<tr>
<td>O-GOAL: Computer Use</td>
<td>C-GOAL: Total Area</td>
<td>Core Border</td>
<td>760</td>
</tr>
<tr>
<td>Important</td>
<td>6700</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>O-GOAL: Security Requirements</td>
<td>C-GOAL: Ambient Lighting Type</td>
<td>P-GOAL: Partition for Manager</td>
<td>Glass Wall</td>
</tr>
<tr>
<td>Medium Security</td>
<td>Flourescent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>O-GOAL: Budget</td>
<td>C-GOAL: Natural/Artificial Lighting Ratio</td>
<td>P-GOAL: Area of Support Staff Office</td>
<td>80</td>
</tr>
<tr>
<td>N/A</td>
<td>0.45</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Plan

<table>
<thead>
<tr>
<th>Case name: Mitsubishi</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Core</th>
<th>Partition</th>
<th>Furniture</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-PLAN: Core Type</td>
<td>P-PLAN: Group Layout</td>
<td>F-PLAN: Furniture for Manager</td>
</tr>
<tr>
<td>Open</td>
<td>One Group</td>
<td>Traditional</td>
</tr>
<tr>
<td>C-PLAN: Space Shape</td>
<td>P-PLAN: Workgroup Division</td>
<td>F-PLAN: Has Task Lighting?</td>
</tr>
<tr>
<td>Angular</td>
<td>Core Border</td>
<td>True</td>
</tr>
<tr>
<td>C-PLAN: Total Area</td>
<td>P-PLAN: Number of Conference</td>
<td>F-PLAN: Shared Workspace</td>
</tr>
<tr>
<td>6700</td>
<td>1</td>
<td>Adequate Space</td>
</tr>
<tr>
<td>C-PLAN: Ambient Lighting Type</td>
<td>P-PLAN: Partition for Manager</td>
<td>F-PLAN: Storage Types</td>
</tr>
<tr>
<td>Flourescent</td>
<td>Glass Wall</td>
<td>Narrow Drawers</td>
</tr>
<tr>
<td>C-PLAN: Natural/Artificial Lighting Ratio</td>
<td>P-PLAN: Area of Support Staff Office</td>
<td>Overhead Shelves</td>
</tr>
<tr>
<td>0.45</td>
<td>80</td>
<td>Pedestals</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outcome</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Goal</th>
<th>Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>O-OUTCOME: Individual Satisfaction</td>
<td>C-OUTCOME: Lighting Quality</td>
</tr>
<tr>
<td>Highly Satisfied</td>
<td>Satisfactory Lighting</td>
</tr>
<tr>
<td>O-OUTCOME: Group Interaction</td>
<td>C-OUTCOME: Air Quality</td>
</tr>
<tr>
<td>Unplanned Interaction</td>
<td>Good Air Quality</td>
</tr>
<tr>
<td>O-OUTCOME: Distraction Frequency</td>
<td>C-OUTCOME: Air Temperature</td>
</tr>
<tr>
<td>Frequently</td>
<td>Warm Temperature</td>
</tr>
<tr>
<td>O-OUTCOME: Accessibility</td>
<td>C-OUTCOME: Partition Cost for Manager</td>
</tr>
<tr>
<td>Highly Accessible</td>
<td>N/A</td>
</tr>
<tr>
<td>O-OUTCOME: Noise Level</td>
<td>C-OUTCOME: Partition Design Aesthetics</td>
</tr>
<tr>
<td>Sporadic Loud Noise</td>
<td>Pleasing</td>
</tr>
<tr>
<td>O-OUTCOME: Privacy-Auditory</td>
<td>C-OUTCOME: Partition Finish Aesthetics</td>
</tr>
<tr>
<td>Low Privacy</td>
<td>Pleasing</td>
</tr>
<tr>
<td>O-OUTCOME: Security</td>
<td>C-OUTCOME: Furniture Cost for Worker</td>
</tr>
<tr>
<td>Medium Security</td>
<td>N/A</td>
</tr>
<tr>
<td>F-OUTCOME: Furniture Total Cost</td>
<td>F-OUTCOME: Adequate Space?</td>
</tr>
<tr>
<td>N/A</td>
<td>Beyond Expectations</td>
</tr>
<tr>
<td>F-OUTCOME: Ease of Adaptability</td>
<td>F-OUTCOME: Ease of Adaptability</td>
</tr>
<tr>
<td>Facilities Adaptable</td>
<td></td>
</tr>
<tr>
<td>F-OUTCOME: Furniture Color Aesthetics</td>
<td>F-OUTCOME: Furniture Color Aesthetics</td>
</tr>
<tr>
<td>Pleasing</td>
<td>Pleasing</td>
</tr>
</tbody>
</table>

### Figure 2:

A Design Case in ARCHIE
Office with fluorescent lighting systems should have diffusers over bulbs, or provide some way of controlling level, to prevent glare.

*Offices with no natural lighting need bright color schemes to prevent them from seeming drab.*

*Highly flexible layout; easy to move offices between different workgroups.*

*Subgroups* (Case Comments)

- 2-3 workers in most offices, high interaction in these subgroups.
- Status expressed in the location/size of offices.

**Case Features Associated with Lighting Quality**

*Design for a wide variety of users.*

*Information Technology.*

Goal Comments

*Not much interaction with workers outside workgroup.*

*Easy to adjust workgroup size.*

Outcome Comments
Figure 4: Part of the Concept Hierarchy
Figure 5:
Model of Lighting Quality
Figure 6:
Clustering of cases based on lighting model.
Using Stories to Overcome Social Obstacles in Design Collaborations

(Extended Abstract)

Eric A. Domeshek
Janet L. Kolodner
Richard Billington
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
email: domeshek@cc.gatech.edu

Craig M. Zimring
College of Architecture
Georgia Institute of Technology
Atlanta, GA 30332
email: czimring@gtri01.gatech.edu

Collaboration In Design

The lone design genius, if not mythical or completely extinct, is surely on the endangered species list. Nowadays, significant design projects require teams of designers coordinating their varied expertise to arrive at effective design solutions. In more progressive circles, design teams do not work alone either. As the world adopts methods such as Concurrent Engineering (CE) and Total Quality Management (TQM), designers are being required to coordinate with customers, marketers, production experts, maintenance staff, and all the other stakeholders likely to be affected downstream by the evolving design. Common sense responses to growing complexity and past failures are crystallizing into new organizational conventions and new design procedures. These new forms reflect a growing realization that higher quality, lower cost products can be produced more quickly if potential problems are recognized earlier in the design cycle.

Despite its desirability, spatial, temporal, and social barriers often prevent effective collaboration on design projects. We are interested in studying the social obstacles to design collaborations. We believe we can build computer tools that will contribute to overcoming those obstacles. In contrast to much work in the burgeoning field of Computer Supported Cooperative Work (CSCW), we are not devising technical and procedural innovations to improve coordination of spatially and temporally distributed

groups. Novel uses of computers, networks, databases, and displays can help deal with the reality that large design teams cannot always work in the same time and place, and may not even work on the same time scale. But when we look at complex large-scale design situations, we see that even once the physical obstacles to collaboration are resolved, there remain social obstacles to turning communication and coordination into effective collaboration.

Teams are assembled, in part, so that their members can contribute a variety of strengths and abilities to the task. A design team is usually composed of individuals drawn from different specialities. They come to the project with different prior knowledge and expectations. They may be working to fulfill different agendas. They fit into different roles, and those roles partly determine how they get to influence the project. These differences multiply as more downstream stakeholders are brought into the earliest design deliberations.

Changes in design practice are beginning to make communication among various participants more frequent. We believe we can help make such communication more effective. If we can directly improve the quality of interactions, we may also have the desirable effect of increasing their quantity as well. We have, for some time, been developing computer-based design aiding tools. Now we want to extend these tools to improve communication and coordination among participants in conceptual design -- to bridge the social gaps introduced by diversity among design participants.

Social Obstacles to Effective Design Collaboration

We see the social obstacles to collaboration as fitting into three broad categories: structural, relational, and cultural. Our group has focused on the cultural differences that restrict people's ability to understand and contribute to a collaboration. We break down cultural obstacles into differences in goals, differences in abilities, differences in knowledge, and differences in conventions. The conventions we are concerned with are primarily distinctive ways of talking about or otherwise representing design issues, but may also extend to conventions for how to proceed with design, or even how to behave in a collaboration. Just as overcoming the obvious physical obstacles to collaboration still leaves all these social obstacles, so too, overcoming the structural and relational problems leaves a complex set of cultural stumbling blocks. Figure 1 sketches our taxonomy of social obstacles to effective design collaboration.

Communication is difficult when parties do not share common background and assumptions, or when they lack a common technical language and facility with external representations. We believe that failures of attempted communication are most apparent, and also most detrimental, in the early stages, during conceptual design. Early in design, the design team -- especially the extended team, including clients and users -- is not a cohesive group. Early on, members do not have much personal feel for one another, and they lack any shared group history or any common reference points. The normal terms of discussion in conceptual design only
exacerbate these problems. In the absence of concrete design proposals, the participants will often talk in terms of abstractions and generalities. But what a structural engineer thinks of as a "large" space may be quite different than what an architect means when he says "large"; the engineer worries about spanning anything from a living room to a convention center to a river, while the architect visualizes something scaled to his image of a building design forming in his head. Similarly, a space described as "large" probably conjures different images for a custodian who has to keep things clean, and the state official who has to pay for the building.

- **Structural**
  - Lack of opportunity to interact
  - Lack of inclination to interact
- **Relational**
  - Lack of trust or respect
  - Differences in status or power
  - Fear of embarrassment or failure
- **Cultural**
  - Differences in goals
  - Differences in abilities
  - Differences in knowledge
  - Differences in conventions
    + Vocabulary
    + External Representations
    + Design Procedures
    + Collaboration Procedures

Figure 1: Taxonomy of Social Obstacles to Collaboration.

By the end of the crucial conceptual design phase, the designers must come to understand what the clients and users really want, while the clients and users must come to understand the design implications of their desires as well as new possibilities created by innovative design. Both of these ends can be advanced by carrying out a dialogue in the context of specific examples of related artifacts. Illustrative examples can be quite powerful; consider, for instance, the difference between abstractions like "we'll have a large waiting area outside the courtrooms" and concrete images of existing courthouse waiting areas that give an immediate feeling for what it is like to be in those spaces.

The Value of Concrete Communication

Our basic premise is that couching design issues in terms of concrete examples is one
of the most useful things we can do to help the various design participants arrive at a common understanding of their problem and the possibilities for solutions. Examples can serve as aids to communication amongst diverse members of an extended design team. For instance, in the context of designing a county courthouse the question of what counts as a "large public waiting area" can be answered by looking at some existing county courthouses and focusing on some particular waiting areas in those buildings that have the character desired. If an architect tries to design a residence for the disabled with standard bathrooms an example telling of how the residents will suffer can make clear to her why her users object. If a client asks for a large floorplate building with private offices all around the perimeter, an example highlighting the dim cavernous interior can make clear to him why his architect objects.

These examples suggest kinds of interactions that could be supported by a design tool capable of storing and presenting evaluative stories along with documentation about existing buildings. Easy access to examples could help design participants working individually to prepare for meetings by anticipating others' reactions. Discussions during actual meetings could be more productive because the team members could easily refer to relevant details of specific examples. Easy access to examples would make it easier to convey the reasons for particular design decisions to other team members. Appeal to examples can also serve to bolster the claims of low status team members who might otherwise be hesitant to advance their point of view.

When contrasted with design guidelines stated as abstract rules, design lessons taught in the context of particular evaluated existing designs have many advantages: examples make design lessons more vivid, and thus more memorable; examples are embedded in a context so users can make more subtle attributions of cause and effect, and thus more precisely determine when the lesson applies; well crafted presentations of examples can also teach aspects of an underlying model of the domain as they teach the higher level design lessons. When contrasted with rule-based critiquing and advising, a story-based approach to representing and teaching design lessons makes sense because many aspects of design lack a solid underlying domain model and because, accordingly, most design guidelines do not really have the force of rules.

**Research Issues for Design Collaboration**

If we expect existing artifacts to influence new designs, it is critical that we create presentations that teach clear lessons and prepare effective external representations of those design experiences. We cannot magically transport a group of users to the sites of sample buildings to experience directly the good and bad aspects of their design; even if we could get them to the sites, we could not ensure they would talk to the right stakeholders there, or that they would be told the relevant stories on cue. Using a particular existing building as an example to teach a design lesson requires crafting a presentation that makes the desired points. The same holds true for most other types of artifacts. We call such presentations **stories**. Stories, in this sense, need not be texts, or need not be solely texts. Our choice of architecture as a domain has
driven us to concentrate heavily on graphical presentations, and we are interested in multimedia presentations in general.

We can take graphic presentations as representative of the sorts of problems we expect to encounter when aiming presentations at audiences with divergent backgrounds. For instance, floor plans are a good way to illustrate some aspects of building design; floor plans are developed and used by architects to facilitate their own thinking and to communicate with some of the other participants in building design. When architects looks at floor plans, certain things are particularly salient and meaningful: they instinctively note the scale, and with their extensive experience, are capable of accurately interpreting the real sizes of the spaces depicted; they can decode the distinctive fill patterns that often indicate something about the materials used for parts of the building; they can interpret schematic symbols or annotations showing construction details. When potential residents of a building looks at a floor plan, they are unlikely to see any of those things; they will simply see the relative sizes and arrangements of rooms. If the point of the story being presented (in part) by that graphic depends on any of the cues that are (initially) invisible to one set of design participants, then there is a potential problem.

We have said that various design participants may not understand each other when each speaks their own native languages, particularly when they talk in abstract and technical terms. Ideally, we will arrive at some way to characterize what kinds of gaps separate the participants' conceptions of the evolving artifact. We will not only describe the differences in their experiences and points of view, but also the similarities. We will develop some insight into which differences are really significant impediments to communication and how similarities can be exploited to compensate. If grounding discussions in concrete examples is to help avoid misunderstandings, we will have to face the question of what kinds of concrete details make sense to different design participants, and what kinds of ways of talking about or illustrating examples will effectively communicate.

**CBR and CBDAs**

We have, for some time, been developing an example-driven approach to helping designers think through the implications of their conceptual design proposals. Now we are looking at how to apply a similar approach to supporting design collaboration. This approach to building design tools is rooted in the Artificial Intelligence paradigm known as Case-Based Reasoning (CBR). In brief, CBR is an appropriate technology on which to base computerized design aids because a significant factor in good design is experience, and CBR is an evolving technology aimed at capturing and retrieving useful experiences. It is a common enough observation in design research that designers in all fields make extensive use of case studies, precedents, and prior personal experience. And it is a common frustration of design practice that finding records of past experience is often annoyingly (or even prohibitively) expensive. Psychological research further suggests that people are often quite poor at picking out
appropriate past experience, even when such experiences are present in their own memories. Our design tools aim to broaden the availability of experiences with design and use of artifacts.

These tools are called Case-Based Decision Aids (CBDAs). They help designers by supplying them with descriptions and evaluations of prior designs relevant to current decision making. In the context of a current problem, an old design can be taken as a suggestion, a warning, or a prod to consider particular issues. For the system to retrieve relevant examples, the user must give it some description of the current problem and any partial commitments towards a solution. In response, the system can present a view of some past design annotated with stories that teach lessons the user should know about in the current situation. The views may be graphical or textual. The stories may be only a selected subset of everything the system knows about the old design. From the initial view, which may encompass only a part of an old design, the user can access other information about that old case. From the selected stories, the user may browse to presentations of relevant design guidelines that suggest how a designer should approach a related class of problems in general. Guidelines, in turn, provide access to other stories (perhaps drawn from other designs) that serve as illustrations, counterexamples or caveats to the general rule.

The issues in designing and building CBDAs include 1) gathering stories and guidelines, 2) preparing effective presentations for a targeted user group, 3) developing a way of describing the stories so they can be found in response to a user's situation, 4) finding an acceptable way for users to express their situations, 5) providing a limited but useful set of presentations and browsing options at any time.

We have been developing CBDAs for several domains, including architecture, lesson planning and airplane design. As we have developed these tools, it has become apparent to us that they could provide a basis for correcting the communication breakdowns so common in large design efforts. We believe CBDAs can play this important role because their major effect is to illustrate the implications of constraints and proposals in concrete form -- as stories of real examples with supporting documentation, illustrations, and analysis.

Doing a good job of supporting collaboration will require further pursuit of several already active lines of inquiry and will result in evolutionary growth in our CBDA's capabilities. We will continue to look for forms of experiential material that can be of use to a working designer, but we will also consider when such materials might be appropriate for other users. We will continue to look beyond case materials per-se to explore the effective integration of experience with generalizations, rules and principles, but we must remember that not all users have appropriate background to understand generalizations (with their technical terms and unstated limitations), or to interpret examples (with their welter of relevant and irrelevant details). We will continue to seek effective organization and presentation strategies, but we will have to consider the background, interests, and media experience of all participants.
Goals of Educational Hypermedia

Our time is often called the information age; there is more and more information around, and its value and importance in everyday life is increasing. To become knowledgeable in general, or to become proficient at a particular skilled task, requires familiarity with large bodies of information. If computers are to achieve their often touted impact on education, an important pair of roles they might fill are those of information distributor and dispenser.

Educational hypermedia systems are a class of computer-based learning environments that assume a significant part of education is giving students and practitioners access to large bodies of relevant information. The “media” aspect of the term “hypermedia” reflects information’s availability in a variety of formats -- as text, as pictures, as sound clips or video, and so on. The “hyper” aspect of “hypermedia” implies that information can be accessed in more than one way, usually in ways that reflect (some of) the inherent structure of the material. The issue for this panel is how best to make use of the possibility of multiple media forms and multiple access methods to produce an educational resource that is really useable.

One thing that distinguishes the set of educational hypermedia systems being presented in this panel, is that they all take considerable care to concern themselves with what kinds of information they present, how and when they present it, and, in particular, the context in which they provide their information. They all assume that information is most valuable when...
presented in the context of its use, and thus they all include, or assume some task environment that engages their user in active problem solving. Starting with models of their users' tasks, they go on to model selected aspects of their domains that matter most to someone engaged in those tasks, and to model some reasonable forms of conversational interaction that could be expected from someone so engaged.

All hypermedia, in its chunking and linking of presentations, embodies some domain analysis and offers the user some choices as to how to proceed. But the systems represented on this panel aim to capitalize on their task, domain, and interaction models to do a better job of putting control of information presentation in the hands of the user. In this short paper, I want to survey how this approach works in the Archie-II case-based design aid for architectural conceptual design.

The Archie-II System

Archie-II is both a design tool and an educational hypermedia system for the domain of architecture. Its purpose is to aid in the conceptual design of buildings -- the very earliest stages of design, when the broad outlines of a building's shape and function are being determined. Archie-II does this by providing access to documentation and evaluation of existing buildings, linked to more abstract guidelines and codes describing good building practice. Presenting the user with examples serves to suggest possible approaches, or warn of potential pitfalls; it raises issues in a concrete way and helps flesh out what would otherwise be abstract recommendations and requirements.

The potential users of Archie-II include practicing architects (most likely experienced designers who are moving into some new genre of building), extended design teams (composed of architects, clients, potential building occupants, and assorted engineering consultants), and student architects. No matter the user category, we expect the user to be engaged in a design problem when they come to the system.\(^1\) The system is used to explore, critique and suggest design options. The user describes his/her problem to the system. The system recalls cases that are similar and presents them to the user. The user then makes decisions about the solution in the new situation based on those cases, updates the problem and solution

\(^1\) In fact, we are currently working to extend the system to include a suite of design tools appropriate to conceptual design. We want to bring more of the active design work onto the computer so that Archie-II is part of an integrated design environment rather than a stand-alone information utility.
specifications, and repeats the process.

The types of information and presentations included in Archie-II are determined by a combination of the domain, the task, and the users. For instance, architectural design is a spatial problem with strong graphical conventions shared by architects; accordingly, the system makes extensive use of design graphics such as floor plans. Conceptual design is only part of the whole design process, so there are certain issues that arise in the context of conceptual design and others that do not; accordingly, the system's examples cover issues such as layout of building zones and circulation, but do not have much to say about choices of finishes or detailed construction techniques. Finally, the very fact that we are trying to aid in a complex open-ended design problem drove the decision to build a case-based design aid, with its emphasis on evaluated prior designs and their relationships to general standards of practice.

Organization of Information in Archie-II

We have established that Archie-II contains a large collection of stories about the good and bad points of particular existing buildings, an accompanying collection of abstract design guidelines, plus graphics and documentation appropriate to illustrate and provide context for these stories and guidelines. Now we must face the question: how can this material best serve the user? We want to avoid overwhelming users with too much irrelevant information. Since they are involved in a design problem when they come to the system, we can expect that some of the system's material is relevant, but much of it will not be. We want them to browse easily through the system, finding only what is most useful to them in the context of their current problem solving.

To this end, Archie-II imposes three main structural systems: the first takes the artifact -- the building -- as the central organizer; the second takes guidelines as the objects being elaborated; the third allows for a fairly rich statement of design issue to serve as the basis for search among the system's stories. I will briefly describe each of these systems in the following paragraphs. The system screen shot in Figure 1 gives an idea of how these access methods appear to the user.

Allowing access to all stories about a particular building from a central browsing point that represents the building itself is a natural enough
organization. One interesting point about the way this is done in Archie-II lies in our adoption and adaptation of conventions from the architectural domain. One presentation of a building is as a set of blueprints documenting the building plans in some detail; in standard practice, these graphics are often fleshed out with textual annotations that communicate additional points not immediately visible in the drawings themselves. In our system, buildings can be presented as simulated sheaves of blueprints, and all of the stories the system knows about a building appear as hot buttons on those blueprints. If the user finds a building they feel is a good model for their current situation they can explore it as deeply as they like by browsing any or all of the stories about that building.

Stories evaluating aspects of particular buildings form the central advice-bearing unit in Archie-II. This emphasis derives from research in case-based reasoning (CBR) which suggests several ways in which concrete stories have an advantage over raw facts or abstract rules. Story advantages include a narrative structure that provides richer context and organization for facts, a concreteness that makes interpretation a simpler process than for rules, and an engaging vividness that contributes to an advantage in memorability. This is not to say that there are no interesting or useful generalizations that could be drawn from, or that could help explain the system's stories; the guidelines in Archie-II are exactly those abstract, rule-like generalizations. But stories are often easier to turn into specific recommendations in similar circumstances than are rules, and a group of related stories can together reveal restrictions on one another, as well as the applicability of any generalization that relates them.

Thus, the second browsing organization in Archie-II allows the user to traverse from a particular story to those guidelines that relate to the issues in the story, and from such guidelines to other stories that help flesh out the real meaning of the guideline's guidance. Stories are related to guidelines as examples of successfully following the guideline's suggested course of action, and as instances showing how violating the guideline can lead to unacceptable outcomes. Stories can also serve as instances where there was an exception to the guideline and another approach worked just as well, or instances where there was a failure of the guideline and an attempt to follow the generalization produced unsatisfactory results. Taken together, the cluster of stories organized around a guideline help make the original abstract statement more concrete. This is clearly a
desirable end, and an extended browsing interaction structured around illustrating an abstract concept is a natural parallel to a useful form of conversation.

The third, final, and main access system in Archie-II supports interest-based browsing. This system recognizes that most interesting stories are neither primarily about a single design issue covered by a single guideline, nor do they bear on an entire building; instead stories are often about interactions between design issues and guidelines playing out in some relevant parts of a building. Interest-based indexing also recognizes that there are other considerations than design issue (or even issue interaction), or building (or even building parts) that a user might employ to designate stories worth browsing. In particular, in keeping with progressive trends in many design and production domains, Archie-II highlights the importance of thinking about issues that arise over the entire life cycle of the building -- from design and construction, on through use, maintenance, renovation, and demolition -- and from the point of view of all potential stakeholders in the building -- including designer, contractor, owner, users of various sorts, visitors and even the community at large. One possible benefit of this extended interest format is that it may help impress on designers that these other concerns need to be considered during design.

In broad outline then, Archie-II’s interest-based browsing makes use of indexes that include the following five attributes:

- **Issue** Goal to be achieved by the building’s design
- **Space** Part of building, defined spatially
- **System** Part of building, defined functionally
- **Stakeholder** Role with respect to building (point of view)
- **Life Cycle** Part of building’s history when issue matters

This index form allows expression of more precise browsing interests than is possible in the first two browsing structures. Matching of these forms is performed by the system at run time, so there are many more potential connections in the system than could ever be established explicitly by hand in advance. A (possibly partial) specification of the five attributes in this index form can be solicited directly from a user as a search key, or can be (possibly partially) inferred from system-observed user behavior. Pairs of these index forms can be used to express an interest in the kinds of interactions that often form the basis for really hard design problems and really interesting design stories.
The plan for the Courthouse &#2, First Floor shows several features:

- **Seperate circulation for the public, staff, and imprisoned parties, privacy and security.**
- **Public Library**
- **Entry**
- **Open Office**
- **System**

**Issue Segregation**
- Real Estate/Contact
- Appraisal Function
- Viewpoint Social

**Space**
- Ownership Purpose
- Sq-Feetage Position

**System Circulation**
- Function Circulation
- Components

**Stakeholder**
- Life-Cycle

**Life Cycle**
- Image 2

**Public Entry to the Judges' Lobby**
- Courthouse Planning Cycle

**SJC -- No Reception or Waiting Area for Judges' Lobby**

Attorneys, businessmen, and other people often have to reach judges or other areas in the building. However, the separation of the private circulation from public circulation makes this difficult. No formal reception area or public entry to the staff area is provided, and as a result, visitors either find their way through unmarked courthouses, branch courts, or other clerical staff to let them in through staff entrances.

The public should be able to visit the judge's private office only by passing through a security checkpoint and then entering the private corridor.

The judge's private office should be adjacent to his secretary's office. The secretary is responsible for screening visitors, so there should be a waiting area for those persons wishing to see the judge. An adjoining reception/public waiting area of approximately 40-45 sq. ft. should be directly located outside the judge's office. It should seat 2-4 visitors and be furnished with comfortable
Summary

This brief discussion of Archie-II emphasized the three major access methods available to a user, and how those derived from knowledge of domain, task, and reasonable interaction styles. Our understanding of the domain and its conventions suggested the use of the "blueprints and annotations" metaphor for building-based browsing. In a deeper way, our understanding of the issues that arise during conceptual design of buildings informed the shape of the index form used in interest-based browsing (as well as the specification of possible fillers for the various fields of that form, which were not discussed in this paper). The very idea of interest-based browsing and a circumscribed list of design issues only makes sense in light of our understanding of the conceptual design task in which the user is assumed to be engaged. The emphasis on efficient browsing likewise stems from an understanding of the constraints on the working designer. Note, however, that our task model does not directly intrude to direct the interaction; this is because design is a complex ill-structured task, and practitioners do not agree on any exact ordering of steps, nor happily accept others dictates with regard to their creative process.

Our understanding of what constitutes a reasonable interaction between a source of information and an information seeker have a more visible effect on interactions. Obviously, the query/response sequence inherent in the notion of expressing interests and searching for relevant stories fits this model of interaction. So, also, does accompanying the presentation of stories with background documentation about the building and context. The linkage of stories to guidelines and guidelines to stories, highlighting four possible relationships between the generalization and the specific instances, further reflects our understanding of how a person would explore the meaning of an abstraction. This arrangement might be compared to the ASK-system conventions for reasonable conversational follow-ups.
Case-Based Design Critiquing

(Position Paper)

Janet L. Kolodner
Eric A. Domeshek
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
email: jlk@cc.gatech.edu

Craig M. Zimring
College of Architecture
Georgia Institute of Technology
Atlanta, GA 30332
email: czimring@gtri01.gatech.edu

There is no end to the intellectual tasks to which AI techniques might be applied, but for many tasks, a fully automated problem-solving system is impractical or undesirable. Some tasks are too complex or require sensitivity to too many hard-to-formalize real world constraints. Other tasks are too critical, or personal responsibility of the human participants is too important an issue. The history of expert systems in medicine shows that systems intended to replace a physician may succeed in solving a (circumscribed) diagnosis or treatment problem, but are far less likely to be accepted than those that advise.

Our own research has most recently focused in the domain of architectural design where we have been faced, most immediately, with problems of complexity. Architecture is a domain in which you can only circumscribe problems so far before they cease to be recognizably architectural design problems. It is not enough for an architect to understand technical systems; she must also understand the social systems her new building will house and shape. It is not enough to understand the economics of construction; she must take into account the history of construction, at
least with regard to her site and region. It is not enough to reason about possible functions of three dimensional forms; she must also consider the esthetics and possible meanings those forms will convey.

Alas, we lack good formal models of how forms constrain functions, and how buildings affect social systems, let alone how forms are judged esthetically and how buildings should relate to context. Theories of technical and economic issues in building design, while not necessarily available, seem more easily constructible. But a system built on such a foundation (aside from having lost much of what makes architecture a broader endeavor than engineering) is likely to fall prey to another form of complexity: formal theories of building design will likely define a search space so large that pushing through to a solution becomes impractical (unless we circumscribe the problem still further; e.g. consider only structural issues, and only those relevant to poured concrete technology).

Given a formal theory that could (in principle) derive a solution to a well stated problem, but that in practice cannot do so efficiently, we might be tempted to drop back and propose to build a solution checker; bidirectional search to connect given start and end points is often easier than searching from one set of givens towards some unknown end. Or, following the lead of the mathematical theorem proving community, we might drop back still further and build solution path checkers; verifying (and occasionally bridging) steps in a solution is much easier than discovering the whole path. Realistically, however, a task like building design is not conceptualized by human designers as chaining of rules in a search space. If users don’t see the task that way, they won’t be inclined to communicate it to the system that way. That makes it hard for a system to check a solution path, and may make critiques couched in terms of faulty reasoning steps less useful than those that refer to problems in the proposed design itself.

If we cannot build a system to solve a problem, and if a human user’s solution is not an explicit solution derivation, but rather, is a (more or less well fleshed out) proposal, then the best thing a system can do is offer a critique at the level of consequences of the proposal. A system intended to offer such critiques would concentrate on recognizing combinations of problem features and proposed solution features which, with some predictability, lead to markedly good or bad outcomes. It would present
those predictive patterns to a designer in a way that would help her to see what changes in a proposed design might be necessary. If done correctly, such a system should be able to bring up issues to a user before they became problems, allowing her to predict with some accuracy the effects of a proposed design early in the design process and before investing much effort in developing the design.

One way of predicting outcomes in complex circumstances is to base predictions on outcomes of previous similar situations. For example, some public waiting areas end up feeling claustrophobic, not due to their size alone, but due to a combination of their size, shape, location, lack of views, style of lighting, and level of traffic. Guesses about whether a proposed waiting area is likely to feel cramped might be based on knowledge about other existing waiting areas.

In an interactive system, this would require recording a set of design situations and their outcomes, analyzing (to the best degree possible) and recording explanations of those outcomes, and making the situations and their analyses available to a designer when similar circumstances show themselves. Outcomes of old similar situations suggest outcomes that might arise in a new situation, helping the designer to focus on issues of relevance and to refine designs appropriately. There is, for instance, a courthouse where the information desk gets overlooked by many visitors because it is set up as a window in part of the lobby not visible from the main entrance; an architect considering the design of the information desk for a new public building might benefit from hearing about the faulty design because it could warn her about the importance of visibility for an information desk and make clear some ways in which a desk might fail to be adequately visible.

The key to being able to make such a scheme work is being able to explain what features in old situations led to their outcomes and indexing old situations according to those features -- when the same feature combinations are seen in new situations, they trigger the recall of old situations that can make relevant outcome predictions.

The cycle of observing situations (explaining and storing them) so that later a reasoner can recall its experiences (retrieving and predicting from them) is largely what Case-Based Reasoning (CBR) is about. The argument
we have developed so far explains why we think CBR is applicable to building critiquing systems. Our group in the Georgia Tech AI lab has been working on applying CBR to design for several years; in the last two years, we have been focusing on the idea of building case-based design aids (CBDAs) -- a class of computer tools that support human designers by prodding them with good and bad examples of design. Our research group, which includes both cognitive scientists and architects, has been developing a CBDA called Archie, intended to support architects during conceptual design. The insights and tools developed in the Archie project are already finding application in a second project where we are supporting early design of jet airplanes.

We have identified six major issues we must address in designing our CBDA systems:

1) how to collect evidence of outcomes (old cases and associated explanations);
2) how to manage the use of cases in predicting outcomes;
3) how to set up human-computer interaction so that a user finds it easy to encounter and interpret these predictions as advice;
4) determining the kinds of advice that should be given at various stages of design;
5) determining what kinds of expertise should be made available, and how knowledge of different types and from different sources should be combined;
6) and finally, managing to make all the different kinds of useful expertise available without overwhelming the user.

As our primary method for collecting interesting stories about existing building, we have adopted the architectural methodology of Post-Occupancy Evaluation (POE). A POE study of a building typically unearths a significant number of ways in which that building fails to satisfy its functional requirements, or, on the other hand, ways in which requirements are notably well satisfied. A well written POE will comment on these issues, noting the data that led to the evaluation and some explanation for the observed effect. One member of our team is an expert in Environmental Psychology whose specialty is the POE method. In constructing the Archie CBDA, one issue is how to streamline the POE process so that more buildings can be studied more quickly. Another issue is how to organize the resulting data so it is easy to access and so that
presentations make focused points.

This second issue feeds into our concern that materials in Archie be presented to the user when they make reasonable predictions about what might happen in the user’s current situation. We address this problem in two ways: first by breaking up the information about any one building into a number of issue-centered lesson-bearing stories; second, by indexing those stories according to conditions under which a user is likely to need to learn the story’s lesson. This means indexing the stories according to the issues they address (including some understanding of who is most affected by that issue, and when in the building’s life cycle the issue is critical), and also according to the parts of the building they discuss (which includes both physically and functionally designated parts). Indexes that license reliable predictions will often depend on relationships between parts (and issues), or on attributes of relatively low level components.

There are two additional approaches we have taken to organizing stories for effective access. First, we connect stories to generalized guidelines that they illustrate. This means that from one story it is easy to find a collection of related stories that flesh out a common point. Second we connect stories to graphical presentations of the particular parts of the particular buildings they describe. This means that from one story, it is easy to find out what else was going on in the same context, should the old building prove to be a good model for the user’s current design problem and proposal. Partitioning building information into reasonable story-sized chunks and organizing those stories for three different kinds of access reflect our concern with the third of our CBDA issues -- arranging a style of human-computer interaction that makes it easy for the user to encounter and interpret advice.

The fourth issue cited above is a recognition that design -- particularly in a complex domain such as architecture -- is an extended process, that different kinds of decisions are made at different phases of that process, and that, accordingly different kinds of advice are relevant at different stages of design. We have concentrated our own work on supporting the early, conceptual phase of design. This is a stage at which it seemed particularly appropriate to adopt our story-based approach to critiquing because detailed causal models are hard to come by for the wide range of relatively abstract issues typically considered during conceptual design.
We have found that only a select subset of the stories turned up by POEs make sense as conceptual design stories. We have been conducting interviews and observations to better determine the range of concerns that rightly belong in a conceptual design aid for architects.

Our story-centered approach to critiquing takes a stand on one form of expertise we believe to be particularly useful in design. But as the inclusion of guidelines indicates, our focus on stories does not exclude some other forms of expertise. In addition to stories (examples), guidelines (generalizations), and basic documentation about existing designs (facts), we expect at some point to include codes (rules) and design principles (heuristics) into Archie. A story-based approach to critiquing does have the advantage of making the multiple experts problem less of an issue than it might be in some other more rule-based approaches. We explicitly expect our guidelines to have exceptions, and in fact, to often be matters of opinion. We expect to have stories that do not satisfy the predictions of some guidelines, we expect to have stories that and guidelines that appear to make conflicting points. That is simply the reality of a domain like architecture, and we expect our users to cope with that reality the way they always do: by fleshing out the general suggestions and specific examples with as much relevant context as possible and by interpreting them with respect to their particular problem.

Finally, if our system is to contain a mixture of examples, generalizations, facts, rules, and heuristics gathered from many sources and often in conflict, we will have to make sure we can manage the user’s interaction with the system to avoid overload. We have invested significant effort in designing an interface and style of interaction that we believe will help us control this potential problem. In particular, by keeping the interaction centered on examples -- stories presented in the context of the building they evaluate -- and by making extensive use of the kinds of external representations that architects are already used to dealing with -- from graphic representation of conceptual relationships to detailed floor plans -- we expect to be able to provide a familiar environment and to control the number of options available at any moment.
USING THE POINTS OF LARGE CASES

ERIC DOMESHEK AND JANET KOLODNER
College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280, U.S.A.

We believe that case-based reasoning (CBR) will prove applicable to design, at least in part, as we have seen designers making extensive use of past cases. Construction of useful systems, however, requires the resolution of many open issues. In this paper we consider three issues in particular: (1) What sort of content should be captured in a design case? (2) How should the content of a complex case be segmented into chunks for use? (3) How should the resulting chunks be indexed for retrieval? These are among the issues we are seeking to address through construction of Archie-II, a case-based aid for conceptual design in architecture. In addition to our approaches to these issues, this paper also discusses our strategic choice to build a design aiding system as opposed to a system that generates designs on its own.

1. Design meets CBR

Design problems cover a wide spectrum from routine configuration of stock parts or parametric variation of existing designs, on up to radically custom creative challenges. Across the board, however, when confronted with new problems, designers tend to make extensive use of old solutions. Consider architecture as a potential exception that ends up proving the rule. When designing the spaces in which we live and work, architects like to think of themselves as artists, and of every building as a unique circumstance. At the level of details this is true enough—no two buildings ever fit the same site, fulfill identical functions within equivalent budgets, and satisfy the same clients; but at a more general level, architects realize they can learn from earlier efforts—many buildings must satisfy similar functions under some shared constraints.

If you asked someone who had no idea of how our judicial system works and who had never seen an existing courthouse to design a new courthouse, you could expect to get a startling, but not a startlingly effective, design. Would the courtrooms provide appropriate seating for Judge and Jury? Would there be separated areas for Plaintiff and Defendant? Would there be ways to display evidence so all four parties could see it? An understanding of function and some knowledge of the forms that have been tried in the past to satisfy those functions is a necessary prerequisite for effective design. This is true not just in architecture, but in design of any kind.

The designer needs to learn about the client's problem and about prior art; unfortunately, this is a time consuming, costly and omission-prone step in design. In architecture, common methodology calls for a special team—the programmers—to gather data on requirements and prepare a program that makes all requirements concrete. The designers then work to satisfy the client's concrete needs as made explicit in the program. As designers work to invent forms that will satisfy the identified functions, they may travel widely to tour existing buildings, survey the scattered literature, or try to relate their problem to famous precedents; if they or their firm have completed similar projects in the past, they may search for ideas in the company files.\(^1\)

This is exactly the sort of bottleneck that modern information technology ought to ease. The catch, however, is that we cannot simply pile data into relational tables, or force given object descriptions into some obvious hierarchical structure. The information we want to manage here consists of recorded design experience and expertise. Figuring out exactly what to store, how to store it, when to retrieve it, how to present it, and even how to collect the information in the first place, are all significant problems.

We believe that a promising approach to these problems can be found in the AI subfield known as case-based reasoning (CBR) (Schank, 1982; Kolodner et al., 1985; Hammon, 1989). As a research paradigm, CBR puts experiences at the center of reasoning and problem solving; our grounding in the

\(^1\) In practice, when the volume within some genre is sufficient, designers and even entire design firms develop specialties. Nowadays, there are plenty of architectural firms that have knocked out dozens of condominium complexes or suburban office parks. Yet it turns out that even in such firms, there is often a lack of corporate memory (C. Zimring, personal communication).
3. What's in a case?

What needs to be stored as part of the record of a case depends on what retrieved case materials will be used for, and why case materials are the sort of thing worth retrieving. For example, in a system that uses cases to support configuration design, case records should note which of several options was chosen for some component, the features of the design problem that led to that choice, and how that component choice fit into the larger design. In a system intended to carry out more creative design, on the other hand, the case records should provide access to more detailed causal models capable of predicting what would happen if the design were transferred to some new situation, and what would happen if the design was transformed to some new configuration. In Archie-II, we are focusing on the following uses for case materials:

• raising design issues;
• proposing responses to design issues;
• identifying pitfalls and opportunities.

Our system presents retrieved materials to the user; the user bears responsibility for understanding and applying (or ignoring) the information presented. We want our system to make these three types of points by telling stories about past designs in a clear and engaging way.

Researchers’ proposals for what should be in a design case range from materials that currently are easily gathered, to materials that may never be available. Design documentation, such as requirements, drawings, parts specifications and perhaps results of some formal analyses, are often generated and stored on-line in structured formats. At the other extreme, the ideal is to have detailed causal and decision models justifying the purpose of each component with respect to requirements, or accounting for the presence of components by appeal to design history and rationale.

In between these extremes there are many possibilities. From collecting the normal records documenting a final design, it is only a small step to storing documentation of earlier design stages including false starts, dead ends and proposals on the path to the final solution. When records are generated on-line, the real trick in holding onto preliminary versions is organizing the information and figuring out the situations in which it would be useful (see, for instance, the ASKJEF system, Barber et al., 1992). The next step is to augment such histories with some rationale for the various attempts and revisions; this is supported to some extent by recently developed tools such as gIBIS (Conklin and Begeman, 1989). Unfortunately, systems that try to capture rationales tend to be intrusive and are resisted by designers who feel they have better things to do with their time than fill out more (electronic) paperwork.

Going to the next step, and representing the purposes of design features and the constraints that ended up determining their values, calls for more elaborate case representations, backed up by causal models (Sycara and Navinchandra, 1989), elaborated as structure-function models (Goel, 1991), or structure-behavior-function models (Gero, 1990). Reasoning over such models is both interesting and useful, but there are some serious limitations. One major problem is simply the amount of knowledge, and the depth of theory, required to build such models.

CBR, on the other hand, is often presented as an appropriate approach to problem solving in weak theory domains. In designing judges’ chambers, for example, we may not know enough about the psychology of perks and prestige to know how many square feet of office space can be traded off against a corner office or a private bathroom, but we can look at old designs and find out how the users felt about them.

This brings us to the kind of case materials emphasized in Archie-II. Archie-II’s cases hold records of existing designs such as blueprints and specifications, but these are augmented with evaluative material that tells us how those designs turned out. Evaluative material focuses on those aspects of a design singled out for accolades or derision. It is collected by soliciting the opinions of those associated with the building—those who are affected by the design process and the resulting building. Consider this example of a story drawn from Archie-II’s current corpus:

The calendar courtrooms in the basement the courthouse are well integrated with the other courtrooms through a secured circulation system; secure elevators provide access to the higher floors which allows these courtrooms to be used for night court proceedings. The calendar courts are also convenient to the public, which has only to go down one floor from the entrance using either stairs or elevator to get to the public lobby connecting directly to the calendar courts.

This paragraph describes one aspect of one courthouse’s design singled out by an evaluator as

To date, the core case materials used in Archie-II have been drawn from post-occupancy evaluation studies—surveys of existing buildings intended to highlight the notably good and bad features of their design. The story above as gathered by C. Zimring, the senior member of the Archie-II project from Georgia Tech’s College of Architecture. While POEs are not universally available, they are reasonably common for some categories of institutional building, and the methodology for collecting such information is broadly applicable.
This principle for breaking building designs into chunks should apply to the design of artifacts in general. Every design domain has its important issues: circulation, security and ambiance; in conceptual design they organize thought and constrain possible solutions. Most complex artifacts are decomposable into some comprehensible way. In conceptual design of buildings these include the problems of circulation within a courtroom as they manifest in some particular part of the building. Ensuring that a story has some issue focus makes sense because a story, to be worth knowing, telling and hearing, must potentially influence a designer's response to some subgoal of the design problem. For example, the first example story above, by focusing on the issues of efficient and secure circulation, can help a designer achieve those ends in a new situation. On the other hand, since a building is large and complex, even a single issue can be more complex than one presentation can bear, so a story may need to limit its consideration of the issue in some comprehensible way. Our sample story does not attempt to say everything about circulation everywhere in the courthouse; instead, it focuses on circulation around the calendar courtrooms.

Cases are broken into chunks by looking for interesting outcomes for design issues in particular parts of the building.

This principle for breaking building designs into chunks should apply to the design of artifacts in general. Every design domain has its important issues that organize thought and constrain possible solutions. In conceptual design of buildings these include circulation, security and ambiance; in conceptual design of aircraft, they might include weight, cost and speed. Most complex artifacts are decomposable into components that often emphasize particular issues and mechanisms for addressing those issues, or attempt to isolate interactions between issues. A spatial decomposition of a courthouse suggests that we can consider the problems of circulation within a courtroom as somewhat separately from the problem of circulation to and from the courtroom. A functional decomposition of an airplane suggests that the engines influence speed by their drag and thrust, while the fuselage affects speed primarily by drag.

Our example stories show Archie-II carving chunks out of a larger courthouse case by highlighting a limited range of issues and discussing how they play out in some part of the building. We believe this is a good fit with the way architects naturally think about their design problems. Though architects may start from lists of functions to be achieved, early in the design process functions are assigned to nascent spaces, and design issues begin to be framed in localized contexts. Our conventions for identifying and writing stories help Archie-II focus its discussion of issues in a way that improves its ability to connect with its users' conception of their problems.

In Archie-II each story is represented by a data structure packaging several sorts of information: (1) a set of text strings (a title, citation, summary and full story text); (2) a bitmap graphic intended to accompany and illustrate the story's text; (3) descriptive information including some indication of the major building systems involved and the major outcome; (4) a list of annotation points locating the story on relevant blueprints; (5) links to other related stories; (6) links to relevant guidelines; and (7) a list of descriptive indexes that provide access to the story in response to user queries (the next section focuses on the form of these indexes).

Note that breaking cases into chunks focused on outcomes with respect to design issues in particular spaces is not equivalent to developing a set of rules for design. The story presentations are intended to preserve as much relevant context as possible, thereby making it easier for designers to understand the described situation well enough to determine its applicability to their own design problem. The guidelines that accompany stories are much closer to traditional rule formulations in that they are abstracted from particular experiences; our assumption, however, is that these guidelines will really only be interpretable in the context of their related stories. Also, the choice of the term 'guideline' reflects our intent to suggest ways of thinking about problems rather than to provide absolute answers.

5. Indexes to design case chunks

In CBR, the indexing problem is the problem of determining the conditions under which old experiences ought to be considered during new problem solving. This is a basic CBR issue, one which all memory-based systems must face. Whether a system is intended to carry out design on its own, or assist a human designer, it must be able to retrieve the right case at the right time—more to the point, it must be able to retrieve the right piece of the right case at the right time.
location of spaces for easy circulation, a designer might want to think about the ambiance of those spaces.

6. Archie-II

Developing the fruits of CBR research into systems that help people by presenting records of past experiences at appropriate times is currently something of a growth industry (Domeshek, 1992; Bareiss et al., 1991). Applying CBR to the construction of decision aiding systems is one important facet of this somewhat larger trend (Kolodner, 1991; Barber et al., 1992). A case-based decision aid, such as Archie-II, is intended to support a user through an iterative cycle of problem refinement: the user describes a problem to the system; the system recalls a case that is similar and presents them to the user; based on the information provided by the system, the user can make decisions about a solution for the new situation and update the problem specification; this updated problem specification serves as a new input to the system, allowing the cycle to continue. The system we are proposing builds on concepts from hypermedia systems, adding a concern for task analysis that shapes the content included and the organization of that content.

The result is shown in Figure 1, which presents a screen shot from our current version of the Archie-II interface. The figure captures the program in the middle of a session browsing through its database of stories about courthouse design; the program is shown in its plan mode, with the large, central pane displaying a floor plan from a particular courthouse. The full screen is broken into eight major panes. The dark pane in the upper left corner is the mode control panel from which the user has selected plan mode. In all five of the system’s modes, the screen is divided into essentially the same eight sections.

---

5 The other major facet is systems aimed at education and training, as described in Schank (1991), Edelson (1991), Ferguson et al. (1991).
in, or affected by, the design. We have also argued that to be useful, documentation and evaluation must be broken up into manageable chunks that can be presented at appropriate times in the design process to teach lessons; case chunks can be deployed to prod the designer into considering important issues, to suggest possibilities, to warn of potential problems, or to point out opportunities. But to serve any of these purposes, it is not enough that case materials be captured, nor that the system be able to identify and manipulate subsets of such materials; it is necessary that the system be able to find the right chunks at the right times. So, finally, we have also argued that an indexing system must be able to express the kinds of issues, approaches, and partial solutions that normally comprise a designer's thoughts-in-progress. The indexing scheme outlined here, with its primary dimensions of issue, space, and system, is an attempt to make those connections possible.

Our conclusions have application beyond architecture. We see the following points as contributions to research on case-based design systems in general:

• organize case materials into short pointed presentations that teach specific lessons based on particular experiences;
• index such stories in terms of the design situations they address;
• describe design situations in terms of design issues associated with particular structural or functional parts of an artifact, and remember to consider issues arising in all phases of the artifact's life-cycle from the points of view of all relevant stakeholders;
• explicitly note interactions between design issues to broaden the user's focus and draw their attention to related aspects of a design with which they should be concerned;
• link stories of specific successes and failures to general guidelines which in turn link back to other related stories in order to allow the user to easily explore a range of responses to the same basic issue.

We believe that it is valuable to work on the issues of case content segmentation and indexing in the context of a complex design domain such as architecture. Facing up to the domain's complexity has forced us to confront several sorts of scale-up problems: the need to represent the many issues, components, and systems that comprise a successful building design; the need to encompass the perspectives of the varied stakeholders over the extended building life-cycle; the need to grow a case library large enough to contain lessons on these many points. We also believe that aiming to build a design aiding system was a key strategic decision that will allow us to successfully tackle this complexity and face such scale-up.

In the end, however, we expect much of what is learned from Archie-II to make its way back into more autonomous systems. To perform the kinds of analyses and make the kinds of decisions human designers do, autonomous systems will need access to the sorts of information human designers rely on—the same case content will be applicable. Further, autonomous systems will need to pick out coherent parts of prior experiences and get access to them when they are relevant—case segmentation and indexing will both be driven by the varied information needs of the design process considered in this work. But we also believe that the products of this research program may find their way into useful application more directly and in the short run; our hope is that case-based design aids like Archie-II will find niches in the everyday real-world design process—in the training of new designers, in the improvement of conceptual design, and in the communication between designers and those affected by their designs.

Acknowledgements

Many people have worked on the development and conception of the system described in this paper. Craig Zimring, our colleague in architecture, and Richard Billington, our research programmer, have been constant collaborators throughout. The Archie-II team also includes Joshua Bleier, Anna Zacherl, Kadayam Vijaya and Ali Malkawi. An earlier version of the system, Archie, was developed with the help of Ashok Goel, Michael Pearce, and Lucas Sentosa; lessons learned from that effort have substantially influenced our current direction. This work has been supported in part by the Defense Advanced Research Projects Agency, monitored by ONR under contract N00014-91-J-4092. All views expressed are those of the authors.

References


Indexing Evaluations of Buildings to Aid Conceptual Design *

Anna L. Zacherl and Eric A. Domeshek
College Of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
{anna or domeshek}@cc.gatech.edu

Abstract

The cognitive model underlying Case-Based Reasoning (CBR) has implications for human performance on many tasks, and the technology developed in CBR research can be turned to enhancing performance. For all case-based systems, effective memory retrieval depends on a successful assault on the indexing problem. But demands on indexes may be different, and in some ways, more stringent, for aiding systems than for autonomous problem-solving systems.

This paper reports on the evolution of an indexing scheme intended to retrieve lessons for architects working on conceptual design of buildings. It illustrates, by example, the process of designing indexing systems, and the demands peculiar to indexes for aiding systems.

Finding Experiences for Experts

It is the norm for expert performance in cognitively challenging tasks to depend on extensive experience. Researchers in the AI paradigm of Case-Based Reasoning (CBR) have been building computational models that account for this fact, and have aimed to produce systems that perform effectively by relying on records of past experiences [Hammond, 1989; Hinrichs, 1992; Koton, 1988; Mark, 1989].

More recently, insights gleaned from a decade of CBR research have been turned towards the problem of building systems that aid humans in performance of real-world tasks [Schank et al., 1989; Ferguson et al., 1991]. At Georgia Tech, we have been focusing on design tasks such as architecture, engineering, and lesson planning [Domeshek and Kolodner, 1992b; Domeshek and Kolodner, 1992a; Chandler and Kolodner, 1993]. We are attempting to produce what we call Case-Based Design Aids (CBDAs) - systems that help human designers by making available to them a broad range of critiqued designs that can serve to highlight important design issues, to explicate abstract design guidelines, and to provide suggestions or warnings about possible design solutions.

Any system that bases its performance on the selective use of items from a large memory must find some way to organize that memory so that the right items can be found at the right time. In CBR, this problem of how to ensure effective selective retrieval goes by the name of the indexing problem, and has long been recognized as one of the key issues in the field [Schank, 1982; Hammond, 1989; Domeshek, 1992]. Research has provided some insight into how this challenge must be addressed, and has produced a sampling of exemplary indexing systems for particular domains and tasks.

This paper reports on the development of an indexing system for one of our CBDAs – Archie-II an aid for conceptual design in architecture. The story we have to tell of the evolution of this indexing system is interesting for several reasons. First of all, there has been relatively little work in the CBR community on indexing systems for physical artifact design tasks. Secondly, because we have been designing indexes for an aiding system we have had the burden and the opportunity to grapple with cases of far greater complexity than is possible when working on autonomous reasoners; the domains in which it is practical to build autonomous systems are necessarily much less complex than those in which humans routinely engage (and in which humans are likely to need aid). Finally, it is an interesting question to what extent the demands of indexing for autonomous and aiding systems may differ; the information available in situation descriptions (which is thus information easily available for indexing

---

*Many people have worked on the development and conception of the system described in this paper. Janet Kolodner and Craig Zimring have shaped this project from the start. Richard Billington, our research programmer, has been a constant collaborator throughout. The Archie-II team also includes Kadayam Vijaya, Ali Malkawi, Ellen Do, and David Brogan. Interviews with architects have been invaluable; the architects included Lane Duncan, Rufus Hughes, Michel Lincolt, and Von Rivers. This work has been supported in part by the Defense Advanced Research Projects Agency, monitored by ONR under contract N00014-91-J-4092. All views expressed are those of the authors.

Presented at the Case-Based Reasoning Workshop
Technical Report WS-93-01, Page26
Indexes to Design Lessons

The first thing to note when considering how to index design cases for architects is that buildings are probably not the right units of storage and retrieval. Buildings are too big and complex to be designed as one piece; instead there are many small decisions that go into a building design, and what would best serve an architect are lessons about which issues are important, how to address these important problems, and what outcomes are likely. Thus, the primary unit of memory in Archie-II is the lesson-bearing story. As an example, consider the following story [Building Diagnostics, 1988]:

The location of the main lobby information desk in the Bristol County Courthouse is inconvenient and makes it difficult for people to find their way. The desk is located in the telephone office, which is off to the left of the main lobby entry. There is a small sign on the telephone office indicating that it is also the information desk, but on first entering the courthouse there is no immediate indication where to go for information.\(^1\)

Since the items being indexed are stories that teach lessons about design issues, it is appropriate that indexes to these stories center on design issues. But issues vary from one part of a building to another, and even pervasive issues, such as efficient circulation may vary significantly in their implications throughout the building. Accordingly, our stories tend to focus on how an issue plays out in some part of a building, and thus our indexes must also specify the relevant parts. Issues also tend to arise at different times during the life of a building: some are important during construction, others during use or renovation. Likewise, not all issues affect all of the stakeholders in a building: some are of most importance to the owners, others to long term residents, and still others to occasional visitors.

With one final distinction, this analysis of features that differentiate issues provides an outline for our indexes. We recognize two different ways of slicing a building into parts: spatially and functionally. Spaces are physically localized building chunks such as floors, wings, or offices. Functional systems such as the electrical and plumbing systems may be distributed throughout a building and are defined in terms of their purpose. So we recognize five primary dimensions as relevant to describing the point of a story with a simple lesson:

- **Issue**: Goal to be achieved by the artifact's design
- **Space**: Part of designed artifact defined spatially
- **System**: Part of designed artifact defined functionally
- **Stakeholder**: Role with respect to artifact defining a point of view on the issue
- **Life Cycle**: Part of artifact's history when the issue matters

A combination of some subset of these descriptors is sufficient to identify a single point that might characterize the lesson of one of our stories (thus serving as a memory label), or might express a user's browsing interests (thus serving as a retrieval probe). But often, a story is interesting because of what it says about the interaction among issues. In one courthouse, the prisoner holding area was located far from the courtrooms, which led to a desirable lack of noise in the court, but also contributed to security problems when the prisoners were being transferred through the building. Stories that address the interaction between issues are best indexed by a pair of the five-featured structures above.

The index outline just sketched, whether used singly or in pairs, does not say much about what sorts of issues, spaces, systems and so forth we will have to represent. It is essentially a road map to the work required for a fully specified indexing system. To give a sense for how such a system is developed, this paper will concentrate on just one of these dimensions: spaces.

Designing an Indexing Vocabulary

Designing an indexing vocabulary is an exercise in exploring the possible descriptions of objects, concepts, and relationships in the domain, and settling on a system that meets several criteria [Kolodner, 1993]:

1. **Relevance**: Index vocabulary must capture those aspects of situations that indicate when one is relevant to another (with respect to a task or tasks). This is just a baseline, common sense guide to help decide what sorts of features ought to be included in an indexing vocabulary.

2. **Extent**: Index vocabulary must be sufficiently extensive to describe an existing or expected corpus of

\(^{1}\)An aiding system looking over a user's shoulder, and trying to explain aspects of the user's problem solving process, likely results in still different sorts of information being easily available for indexing.

\(^{2}\)Note that Archie-II's stories are always shown with a presentation (usually graphic) of the artifact being discussed, and are generally accompanied by an illustration that amplifies the point being made in the text.
memory items for the range of intended uses. We want to cover a large corpus of design stories, and eventually would like to allow users to enter their own stories.

3. **Specificity**: Index vocabulary must get specific enough to make all useful discriminations among items in memory. We want to retrieve only those stories that are most relevant to a designer's situation.

4. **Generality**: Index vocabulary must also contain components general enough to capture relevant similarities among the items in memory. We want to be able to retrieve stories whose indexes are inexact matches to a designer's situation, if that is the best there is in memory.

5. **Usability**: In an aiding system, it is helpful if the indexing vocabulary corresponds closely to practitioners' conceptions of their domain and task. We want designers to feel comfortable using the system and we want to minimize the amount of inference required of the system at retrieval time.

Actually, extent, specificity and generality are all closely related. What it means to cover a corpus (to have sufficient extent) is just to be able to note the similarities and differences between the items in the corpus. Any (relevant) feature introduced into an indexing vocabulary is likely to improve the system's extent, and may serve both specificity and generality; the feature will appear in some indexes but not in others, and thus will discriminate; but for all indexes in which the feature appears, it will be capturing a similarity.

An architectural design aiding system presents serious challenges in balancing the criteria. To be useful, we must build a large corpus, we must index it using terms an architect might naturally employ, and we must build in sufficient flexibility to satisfy the often idiosyncratic approaches of many different architects. All this must be accomplished while not burdening the user (or the indexer) with too many choices.

Satisfying all the criteria in any interesting domain is actually a hard design problem. Arriving at an acceptable solution generally requires an iterative process of analyzing the domain and task, proposing index components, and evaluating those components with respect to the criteria (which may involve actually building partial memories based on those proposals). We have been following this process for Archie-II's indexes, and are currently on the third major loop through the cycle for the space components. The next section describes the history of that design process with particular attention to the rationale for our current solution.

**Spaces as Index Components**

The possible spaces in a building are many and various. There are also many possible ways of describing those spaces. Here we summarize three approaches considered for the Archie-II system.

**Spaces: A First Pass**

The first attempt at an indexing vocabulary for spaces was developed, in part, to suggest the way such attempts might proceed in general. Based on a corpus of stories we then had in hand, a list of quite specific space types was created; since our cases were drawn from courthouses, this list included such items as parking-lot, entry, lobby, information-desk, vestibule, and courtroom. Just as we characterized our initial index proposal in terms of a set of five dimensions, we proceeded to pick out dimensions (in this case four of them) that together began to characterize the spaces in our list.

The four dimensions selected to describe spaces were ownership, purpose, size, and position. Ownership encoded common patterns in who used the space. Purpose was intended to indicate what the space was used for. Size was specified in terms of square footage. Finally, position encoded the space's location within the building.

For each dimension we then proceeded to specify a set of possible fillers. In principle, this system was not limited to the original list of named spaces – any combination of the defined fillers for any subset of the dimensions could be used to describe a space. In practice, these dimensions and the fillers provided for them were not even sufficient to do a good job of describing the spaces in our original list. For instance, we could not distinguish criminal from civil courtrooms.

Still, this initial proposal had some positive features. Because it allowed use of a set of everyday names for spaces, it was relatively easy for an architect to use. The interface problem created by a potentially lengthy list of names to choose from was somewhat mitigated by the ability to enter a partial description of a space (by choosing fillers for any of the dimensions); given such a partial description, the system would then prompt the user with a limited menu, including only those spaces that satisfied the specified conditions. Of course, here the limited expressivity of the space description language became a problem, as did the fact that choosing fillers for the dimensions was not as intuitive as the straightforward choice of space names.

Limiting the characterization of spaces to the four dimensions and their fillers not only affected usability, but also failed the tests of extent, specificity and generality. The following example illustrates a failure of specificity. Imagine an architect is designing an office suite for a large accounting department and is concerned with appropriate use of natural lighting. Currently, Archie-II only stores courthouse stories, so while it has stories about lighting in several office spaces, the names associated with the spaces include "probation office", "judge's lobby", and "magistrate's office", but not "accounting department". Now, by choosing from among available values for the purpose and ownership dimensions the user could tell the system that the space she is concerned with is a work
area generally off limits to the public. On that basis, the system might retrieve stories associated with any of the courthouse office spaces mentioned earlier. But consider that of all those spaces, there is one that is clearly most similar to an accounting department; the probation office is an office suite that includes a large private office for the manager, a group of smaller offices for staff, and a place for a receptionist. Other courthouse office spaces do not house the same number of people, nor break up the space with the same sorts of dividers. When the system retrieves lighting stories from all the courthouse offices, the user is forced to sift through a large collection of stories, most of which are unlikely to be relevant to her lighting problem.

Spaces: A Second Pass

In our second attempt at a vocabulary for spaces our primary concerns were to increase the expressivity of the language and to take better account of the features that mattered during conceptual design (and thus that would determine the relevance of a story’s lesson). Among the constraints architects face when designing spaces, some of the most powerful stem from who will use the space, what they will use it for, and what kinds of support are required for such use. For example, a room where the primary activity is discussion must be set up differently than a room where the primary activity is lecturing; a room for discussions by groups of 20 must be designed differently than a room for discussions by 3 or 4; a typical manager’s office must function not only as a small group discussion room, but also as a place for private desk-work.

In a sense, answering these questions would force us to expand on the original dimensions of ownership and purpose. A separate effort was anticipated to capture more of the physical attributes of spaces (in effect, expanding beyond the original pair of dimensions, size and position); note that this second extension would more likely address the shortcomings identified in the office lighting example of the last section. The focus on issues critical to conceptual design also led us to introduce in this second pass an important set of features that had not been considered at all the first time around. Often a space’s design is strongly influenced by its interactions with other spaces. So while focusing on uses of spaces, we also began to look at the relationships between uses of separate spaces.

As a way of encoding descriptions of the people, activities, and props associated with a space we adopted the script formalism [Schank and Abelson, 1977]. Under this proposal, the space dimension of our five-part index outline was to be filled with a set of scripts and references to related spaces. Similarities among the specific activities, role-fillers, props, and spaces would help determine the system’s judgments of similarity between story indexes and user queries.

Below is an example of a script-based space description for a court clinic (a court clinic is where a psychologist or social worker counsels young probationers):

- **Court Clinic:**
  - **Role:**
    - Psychologist / social worker
    - Juvenile probationer
  - **Props:**
    - Desk
    - Toys / toy storage
  - **Activities:**
    - Psychologist talks to probationer
    - Probationer talks to psychologist
    - Psychologist works at desk
    - Probationer plays with toys
  - **Related Spaces:**
    - Waiting room
    - Juvenile probation office
    - Conference room

The kinds of information included in this space description are, in fact, important to architects; our second pass at index design improves on the original scheme by significantly broadening its extent. Unfortunately, this approach swings too far in the direction of detail. Users will not put up with having to constantly be explicit about much that is normally left as tacit knowledge about a design problem, so usability has actually deteriorated. What makes this more than just an interface problem is that there still remain important similarities and differences among spaces that cannot be described in terms of scripts and related spaces (or in terms of some improved vocabulary for physical description).

Consider the constraints on a juvenile courtroom. A description of what goes on in a juvenile trial will not really be able to capture the important notion of confidentiality, and a design for a juvenile courtroom will not succeed without taking confidentiality into account. The fact that court proceedings are supposed to remain confidential is actually a matter of what does not happen: someone not associated with the trial does not get to observe it. Scripts do not normally include such negative statements, and even if they did, requiring a user to express such a basic concept in such particular terms would be awkward.

Finally, we note that in addition to excluding some important features, the scriptal approach to describing spaces may include many features that do not contribute much to discriminating stories in our corpus. For example, while the need for props such as a desk, file storage, chairs, and tables in a space for office-work places some constraint on the space design, when dealing with a building like a courthouse, where many of the spaces share that feature, we are not getting much return for our descriptive effort.
Spaces: A Third Pass

The first attempt at an indexing vocabulary for spaces captured only a small number of the relevant features, and the second attempt, while capturing more, still neglected many important abstractions and required too much attention to detail. Our strategy for arriving at a system with the right amount and level of detail was to go back and pay more attention to the way architects do the work we aim to support.

Accordingly, we have been devoting more effort to studying architects' processes of conceptual design. We need answers to the pair of questions: What kinds of decisions do architects make during conceptual design, and what features of the design problem and its evolving solution do they use to make those decisions? So far, we have a preliminary answer to the first question that breaks down conceptual design issues into two major categories: the organization of spaces and the features of individual spaces. The features of the spaces break down, in turn into three categories: the people and things that play important roles in the space, features of the exterior, and features of the interior. So in all, we have four clusters of features to consider.

There are four components to space organization:

- O1) Relationships between spaces
- O2) The strength of inter-space relationships
- O3) Distances between spaces
- O4) The orientation of spaces to the site

Roles fit into three categories. Primary and secondary roles indicate who is using a space, differentiating between those who use the space frequently and those using it less frequently. Props are inanimate objects associated with particular spaces. These then are the three role components:

- R1) Primary roles
- R2) Secondary roles
- R3) Props

The exterior can be described along four dimensions:

- E1) Openings (such as doorways and windows)
- E2) Materials
- E3) Three-dimensional shape or form
- E4) Space flow (paths between spaces, and the 3-D shape of those paths)

Finally, interiors can be described using three dimensions:

- I1) Function
- I2) Materials
- I3) Space characteristics (such as size, lighting, and thermal comfort)

This outline of conceptual design issues serves as the basis for space descriptions in our indexing vocabulary. Most of these features apply to a space; some apply to relationships between a space and some other space. When describing any particular space (considered the focal space) we characterize it using both sets of features. The features that characterize the space's relationships to other spaces implicate two other sets of spaces: included spaces cover parts of the focal space (differentiated based on function); related spaces are other disjoint spaces with interesting relationships to the focal space. The included and related spaces can in turn be described using these kinds of features. Applying these features to a focal space and its relationships to other spaces results in the following form for space descriptions:

- **Focal Space**
  - O4 - Orientation to site
  - R1 - Primary Roles
  - R2 - Secondary Roles
  - R3 - Props
  - E1 - Exterior openings
  - E2 - Exterior materials
  - E3 - Form
  - I1 - Function
  - I2 - Interior material
  - I3 - Space characteristics
  - **Included Spaces** [Pointers to other spaces]
  - **Related Spaces** [Pointers to other spaces]
    - O1 - Relationship type
    - O2 - Relationship strength
    - O3 - Distance
    - E4 - Spaceflow

Most features apply directly to a particular space, and are shown here associated with the focal space. Note that included and related spaces are themselves spaces that can, potentially, be described by the same features. A smaller set of features (O1-O3 and E4) bear directly on relationships between spaces, and these are shown here nested beneath the related spaces.

At any given point in the design process, the architect is most directly concerned with some particular level of analysis which defines what count as focal spaces and what count as included spaces. For example, during conceptual design of an entire courthouse, a likely unit of analysis is the judges' lobby, consisting of judges' offices plus associated support areas such as clerks' offices and private restrooms. At the point where the judges' lobby is a focal space, we expect most of the relationships of interest to be expressed as relationships between the lobby and other areas, such as the courtroom. This contrasts with another, finer level of analysis, at which relationships might be noted between included spaces such as a judge's office and the judge's bench in the courtroom.
We illustrate this new approach by describing a jury room as part of a query about juror circulation patterns to and from the room. The architect knows that jurors play a primary role not just in their jury deliberation room, but also in courtrooms and in the jury pool room (where jurors are selected for trials). If the architect is concerned about how to arrange circulation to and from a jury room, the following description of that room can serve as a useful piece of a query for relevant stories:

### Focal Space

<table>
<thead>
<tr>
<th>R1</th>
<th>Primary Roles</th>
<th>Isolate primary role</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Jurors</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>13</th>
<th>Space characteristics</th>
<th>Isolation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acoustics</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Included Space</th>
<th>Meeting Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Included Space</td>
<td>Private Restroom</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Related Space</th>
<th>Pool Room</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Related Space</th>
<th>Relationship type</th>
<th>Relationship strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1 Relationship type</td>
<td>Same primary role</td>
<td></td>
</tr>
<tr>
<td>O2 Relationship strength</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Related Space</th>
<th>Courtroom</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Related Space</th>
<th>Relationship type</th>
<th>Relationship strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1 Relationship type</td>
<td>Same primary role</td>
<td></td>
</tr>
<tr>
<td>O2 Relationship strength</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

In this description the jury deliberation room is the focal space with respect to which the architect is concerned about circulation. The architect specifies only a few aspects of this room: its primary residents will be jurors, and its primary function is to keep jury deliberations confidential (which is to be accomplished by physically isolating the jurors from other building users and by making sure that the room is acoustically isolated from other spaces). The two included spaces describe a logical guess by the architect: there should be a private restroom. In this context, the architect specifies no further details of the included spaces.

The space description above would be accompanied by other index components to indicate the architect’s interest in stories about the implications of jurors’ circulation to and from the Jury Room during normal use of the building for trials. Given this description, the system could have a basis for choosing the following warning story describing circulation patterns for jurors moving between the related spaces that fail to ensure desired isolation:

*On the second floor, the superior court and municipal court jury assembly rooms are placed around a service core that serves both of them. This core contains direct access to the stairs and elevator used by the staff. This can give jurors an opportunity to take the elevator to other floors and make contact with people in the staff areas, which are supposedly segregated from the public.*

This system, based on modeling the indexes after the actual decisions made during conceptual design, does better at satisfying our criteria than did the previous attempts. Extent, specificity and generality are all improved by attending to a wider range of features that are attended to by architects. This scheme is not as susceptible to the sort of failing we saw with our first proposal, when the system could not notice how an “accounting department” was more like a “probation office” than a “judges' lobby” or “magistrate's office.” Despite including descriptors such as the role filler “juror” this scheme also need not be unduly bound to the idiosyncracies of the courthouse domain; we have preliminary breakdowns of such categories in terms of underlying attributes that capture their relationship to the building and its spaces. We take it as a constraint on the vocabulary items we posit that they not only contribute to distinguishing among our courthouse cases, but that they also contribute to identifying distinctions likely to matter in other architectural (and even other design) domains.

### Bubble Diagrams and a Bubble Editor

Hewing to architects’ own distinctions and vocabulary as done in the third pass, should improve expressiveness. Unfortunately, the complexity of this query format is likely to raise serious problems with usability. We are therefore trying to provide a reasonably intuitive way for architects to designate spaces of interest. Architects love graphic representations, and are adept at visualizing spaces. It would be nice, if we could allow our users to pose queries simply by pointing at representations of spaces on screen. The problem is that there is no canonical graphic representation for all the features of spaces relevant during conceptual design (and, it would premature to draw detailed CAD diagrams at such an early stage of design).

It turns out that architects have, however, developed graphic forms appropriate to conceptual design: many architects develop their early ideas using bubble diagrams. In a bubble diagram, a bubble (a blob that does not necessarily represent shape but does represent size) is drawn to represent each space. Lines are drawn between bubbles to indicate relationships: dark, thick lines represent strong relationships while light, thin lines represent weaker relationships.

We intend to provide users of Archie-II with a bubble diagram editor for creating and viewing these
schematic design aids. Ideally this tool will comfortably support their normal conventions but will also make it easier to capture the features we need for query processing. When a bubble is drawn in the editor a template will pop up on the screen asking the user to fill in the blanks for the various features of the space. When a line is drawn between bubbles another template will pop up asking the user to describe the relationship between the two spaces.

The bubble editor will assist the architect with conceptual design in three ways. The first way is simply to improve on a representational technique they are already using by automating editing and preserving an on-line record of their work. The second way is by helping the architect describe features of the spaces. The third way is by helping to form queries to the case-based design aiding system.

Conclusion and Future Work

This paper has focused on two issues. The first was explaining and illustrating the iterative process of designing an indexing system. The second was arguing for the appropriateness of the current system for indexing stories that teach lessons about conceptual design in architecture. By describing stories using features that architects consider during conceptual design, the indexing system should naturally provide users with an easy way to specify probes to the case-based system.

It is interesting to study both the evolution of our indexing system and its current state. Everything seemed to fall into place once we focused on basing the index vocabulary on the decisions made during conceptual design. This made the vocabulary familiar to the architect and opened up opportunities for graphical representations. It seems like common sense to base the vocabulary on the decisions made, but it is sometimes useful to have the obvious stated. Although we have not made the attempt yet, we believe that this approach is transferable to conceptual design in other fields.

Further studies with architects, including interviews and observation, are planned to help us refine this approach still further. We hope to learn more about what space characteristics are important to architects during conceptual design. We also hope to learn more about how architects use bubble diagrams, since they appear to be a strategically impoverished representation that force focus on those central issues that matter early in design.

References


The Science Education Advisor:  
A Case-Based Advising System for Lesson Planning

Dr. Terrell N. Chandler & Dr. Janet L. Kolodner  
Georgia Institute of Technology, College of Computing  
Atlanta, Georgia 30332-0280, (404) 894-7435  
jlk@cc.gatech.edu, tnc@cc.gatech.edu

Key Words: Science Education, Intelligent Multi-Media System,  
Knowledge Representation, Knowledge and Skill Acquisition

Abstract

The Science Education Project (SCIED) is a case-based hypermedia browsing tool to aid elementary school teachers with science lesson planning. The program provides several different forms of advice to teachers: lesson plans for a variety of topics, guidelines for dealing with difficult teaching situations (e.g., how to group students appropriately or what to do when an experiment or demonstration doesn't work), and the experiences of experienced teachers in making the lesson plans and teaching strategies work. Advice is readily available to teachers while they are browsing through the program for lesson ideas or looking for strategies that will help solve a problem they are having with a particular lesson. In effect, the computer system is a forum through which teachers and researchers can communicate their experiences of what works in the classroom and why. Informal evaluation by teachers and curriculum developers have been very positive. A formal evaluation of the system will be conducted this summer and the results presented at the conference.

Two years ago, we proposed the use of case-based decision aiding (Kolodner, 1991) as a methodology for creating a system to advise teachers. Case-based decision aiding is based on the case-based reasoning paradigm. A case-based system augments the memory of a person by making the experiences of others available. In 1991, our system, then called AI-ED, was in its infant stages. While we knew that case-based reasoning provided a good base for building such a system, there were many issues that we knew we still needed to address, including the content of the case library, the representation of cases, the indexing of cases, and the creation of an interface that would be easy to use. Since that time, we have interviewed teachers, finding out what is hard for them, what they need help with, and collecting their experiences. This analysis has allowed us to begin to address the issues brought up in the earlier paper. In this paper, we report our current solutions along each of these dimensions and present SCI-ED, our current implementation.

Presented at the International Conference on AI and Education, Scotland, August, 1993.
Introduction

The SCIED advisor (SCIED) is a case-based hypermedia browsing system for collecting and dispensing of ideas and advise for the teaching of science in elementary school. Our approach is to apply what the research community has learned about cognitive functioning to build a hypermedia browsing system which: a) augments the memories of teachers, b) advises teachers on how to best apply a lesson or pedagogical approach to their class, and c) promotes communication and collaboration within and across the educational and scientific community. Based on the teachers stated interests or needs SCIED suggests candidate lessons for the teacher to adopt. It also identifies problems and gives solutions for instruction in content areas (e.g. issues involved in the teaching of chemistry), in the science process (e.g. getting kids to measure accurately) and in promoting a positive learning environment (e.g. how to group kids for cooperative learning). In addition to providing lessons and abstract solutions to teachers' problems, SCIED provides teachers with advice through concrete guidelines and "war" stories from teachers who have used the lesson before.

Two years ago, we proposed the use of case-based decision aiding (Kolodner, 1991) as a methodology for creating a system to advise teachers. Case-based decision aiding is based on the case-based reasoning paradigm. In case-based reasoning, a reasoner makes decisions based on what worked and did not work in previous situations or "cases". Cases play several roles when remembered. They can be used to suggest a means of solving a problem. They can warn of the potential for an unanticipated problem to occur. They can help in critiquing and repairing an already worked-out plan.

A case-based decision aiding system stores a library of cases in its memory indexed by features that predict their usefulness and makes those cases available to users as appropriate. It works interactively with a human problem-solver, augmenting the memory of the human user with the experiences of others and providing the human with cases to use in reasoning. It provides a workbench for the flexible creative decision making of the human user.

The technology, we said in 1991, seemed particularly suited to helping teachers plan lessons. While books and manuals give guidelines, they do not tell teachers exactly what works and doesn't work and what to watch out for in the classroom. This technology is aimed at giving teachers more concrete advice in the form of the concrete experiences of others.

In 1991, our system, then called AI-ED, was in its infant stages. While we knew that case-based reasoning provided a good base for building such a system, there were many issues that we knew we still needed to address, including the content of the case library, the representation of cases, the indexing of cases, and the creation of an interface that would be easy to use. Since
that time, we have interviewed teachers, finding out what is hard for them, what they need help with, and collecting their experiences. This analysis has allowed us to begin to address the issues brought up in the earlier paper. In this paper, we report our current solutions along each of these dimensions and present SCIED, our current implementation.

We begin with a simple look at how a teacher interacts with the system.

**The Oyster Shell lesson.**

The first thing a teacher does, when using SCIED, is fill out a one page form that gathers information about the teacher's class, what his or her objectives are for the lesson and/or any areas in teaching where they are having trouble. For example, the class may have 24 5th graders in a classroom with no sink. The teacher may indicate that he is teaching about circuits; he indicates that he is having problems with students not recording precise observations during experiments, and that he would like some tips about grouping kids for cooperative work.

The browser uses this specification to retrieve the appropriate information for the teacher. Paths leading to that information are highlighted for the teacher to follow. In this case there are three paths. One path leads to a screen where suggestions are made for lessons on circuits that are appropriate for 5th graders. Another path leads to strategies for grouping kids in a cooperative learning situation (see figure 1). A third path leads to strategies for how to train children to make precise recording of their observations. One strategy presented is to have the students practice observation and recording early in the school year; the "Oyster Shell Activity" is suggested as a good activity to use for this purpose.

In part I of the oyster shell activity (see figure 2) students arrive in class to find two piles of oyster shells. They are instructed to take one shell, observe it for 60 seconds, and then place it back in the pile. After the teacher mixes up the piles, students try to find their original shell. This is supposed to impress upon them the need to make careful observations. In part II the students are given a 3 X 5 card and instructed to describe their shell so that someone else can find it. The shell are returned to the piles. Cards are exchanged between students and students try to find the other person's shell using the card.

The author of this lesson has several recommendations for how to make this exercise most effective. Throughout the description of the activity is highlighted text indicating that there are stories and guidelines associated with the text. For example, the materials section is highlighted. The associated guideline warns the teacher to make sure the restaurant, where you pick up the shells, washes them otherwise they really smell. Another guideline warns the teacher to "watch that students don't mark
their shells because students can be very clever...". If the teacher wants to see more about that guideline, a story about kids adding marks to their shells is presented. In part II of the activity, one story tells of an opportunity. Inevitably some of the brighter kids ask whether they can draw a picture or trace the shell. The system points out that teachers can use this opportunity to talk about the important role diagrams play in keeping precise records.

**Knowledge Representation in the SCIED Advisor**

**Content of the Case Library**

In 1991, we believed that the library of our system should hold cases composed of the experiences of teachers in the classroom. We said then that we needed to find out which were the most important kinds of experiences to put into the case library to encourage the creation of good lesson plans.

We have learned, from discussions with teachers and curriculum coordinators, that cases (i.e. experiences) by themselves are not enough. Cases need to be imbedded a concrete plan of action, such as, a lesson plan. This led us to investigate what plans of action would lead to contexts where advise to teachers would be useful.

We discovered three "plans of action" context:

1. developing lesson plans and units
2. choosing between different approaches to teaching
3. Developing strategies to deal with "issues" concerning organization and management of science classes

As a result of this analysis SCIED associates its cases with these three "plans of action". Lessons and units are concerned with what should be taught to the student. Lesson plans are the sequence of what goes on in the classroom over a single session, units hold lesson plans together. Approaches and strategies are concerned with what the teacher should know. Approaches inform teachers of teaching methods that could be applied across many content areas. Cooperative learning is one such approach to teaching; the lecture is another. Strategies help teachers with practical problems they face when conducting a science lesson. Assessment and evaluation, motivating interest, or dealing with failure are three examples of problem areas where strategic advice is sought by teachers. All three "plans of action" are related to each other. Certain approaches may be used to facilitate learning a concept or skill; certain strategies may need to be integrated into a lesson plan to address organizational issues that arise during that lesson. The case library captures these connections.

Cases are the experiences of teachers engaged in carrying out a lesson plan or using a teaching approach or strategy. Cases take the form of guidelines and stories, and can be associated with one or several lesson plans, teaching approaches or strategies.
Given that the library needs to hold these many kinds of items, we now have a broader problem than simply identify which kinds of cases to put in the case library. Now we also have to identify which lesson plans, approaches, strategies and their associated cases belong in the library. To tackle this problem we framed it in terms of what issues need to be addressed by a lesson, an approach or strategy to successfully teach science. Three broad areas where issues consistently arise are: the teaching of science content, the teaching of thinking skills, and classroom issues (independent of what is being taught). To make the collection task manageable we have narrowed our focus to address a subset of the issues in each of these areas. These are: the teaching of physical science concepts, the teaching of science process skills and classroom issues around cooperative learning, evaluation and recovering from potential failure.

Representing & Indexing Cases & Other Items in the Case Library

Our 1991 paper listed the representation of cases as an important research area. In SCIED, cases and other items (e.g. lesson plans) are presented to users in a form that makes them easy for a person to understand. It is necessary, however, for the internal representation to capture enough about each item so that the system knows when to display it. Each type of item that can be presented has different representation requirements. The essence of a lessons, for instance, is described by its objectives, teaching method, activity context (e.g. materials and equipment needed for the lesson) class context (e.g. students' age, class size) and guidelines and stories.

The description associated with each item control the circumstances under which cases or lessons are retrieved, thus acting as indexes descriptive vocabulary is arranged in a series of hierarchies that allow retrieval algorithms to retrieve partially matched cases given a probe. For instance, two lessons may have similar but not identical objectives. One lesson objective may be teaching skills for making precise observations; the other may be teaching skills for precise recording of observations. If the teachers specifies a need to teach recording skills both lessons may be retrieved.

The content of indices and how they are arranged with respect to one another continues to be a major research problem. For the advice to be acceptable to the teacher, the system must provide the right advice at the right time and place. On the one hand it would not be appropriate to present a lesson to a kindergarten teacher that is targeted for a 5th grade class. On the other hand, it is important to recognize when advice given in one area is appropriate for presentation in another. In the oyster shell lesson, for example, the teacher who submitted that lesson cautioned that the shells should be washed before they are used. The system should recognize that this case is applicable any time a lesson uses shells as an integral part of an activity.
Implementation

There are two phases of the system where the indexing scheme is used. It associates items with each other at entry time and it is used to choose appropriate items for display while browsing. For example, if a new lesson that is entered lists "shells" as part of its materials; the system may present the Oyster Shell Activity case about cleaning shells as a candidate guideline appropriate for this lesson. If the user accepts the guideline as appropriate then a hard link between the lesson and this guideline is created.

During browse mode the system uses the teachers initial specification as a probe to follow the links of the index network. If a lesson is linked to a node teaching something about electric circuits, and another node having to do with cooperative learning then it will be rated as more relevant than others lessons that fit only one of these criteria. While the connections between informational units are hard linked for fast retrieval, soft links are used for presentation of the information. Different heuristics can be used to determine under what circumstances it is most appropriate to highlight important information or hide inappropriate information.

The Interface

Through our work over the past year we have found that for a system, such as SCIED, to be an effective advisor, the interface needs to be structured so that the logic of its operation is self evident to the user and the information presented is pertinent and useful (Kolodner, Billington, Chandler, Domeshek, 1992). Several issues have to be addressed in order to achieve this goal: what specific information should be included in presentations; how should the information be structured for presentation; how should presentations be related to one another, and, how should the user navigate through the system.

The answers to these questions are guided by four types of models: task models, user models, domain models and interaction models. A task model represents the steps or components of the design process which we are assisting. The user model enumerates reasonable assumptions about a user's knowledge and expectations. The domain model tells us how the artifact or plan being designed is described by the designer. Finally, the interaction model suggests metaphors that can help shape the navigation mechanism and influences the form in which cases are presented. The interaction of these four models are what determine the shape of the interface.

The domain model for SCIED indicates there are three overall concerns a teacher has in constructing a lesson plan: the content area; the skills being taught to or required of the students; and pedagogical concerns, like cooperative versus individual learning, learning from failure, etc. A teacher will likely want to get
some information about a variety of these topics which seem particularly relevant. Therefore, one set of screens are organized to be about particular topics in each of these three major areas. These screens use an interaction model of having a forum of experts available for consultation (see figure 1: cooperative learning). Part of the navigation mechanism is to explicitly show to the user this organization based on the three overarching concerns for a lesson designer. The task model suggests a teacher also wants to see actual lesson plans for specific ideas to include in their own plan (see figure 2: The Oyster Shell Activity).

Through the interview process we have found that the best teachers practice contingency planning while they develop their lesson plan. To help less experienced teachers SCIED is designed with contingency planning as part of its task model. A contingency planner creates a plan and then projects the consequences of that plan so that it can evaluate what aspects of the plan could be improved. The process takes into account potential problems and opportunities that might arise. It usually results in either fixing a plan or in creating alternate paths through a plan that can be carried out if warranted. The interaction model has the teacher, who developed the lesson, present her contingency plans through guidelines and stories. The user model says that teacher's aren't experts in every content area they have to teach nor are they expert about how to teach every skill they need to. Hence we include background information in each lesson to bolster their knowledge of what they're teaching.

Conclusion

The emphasis for educational change in this project is on the teacher. If any real sustained pedagogical change is to be realized, then a support structure will need to be built to facilitate the growth of scientific literacy of our teachers. A dynamic storage and information retrieval technique, such as CBR, underlying a hypermedia browsing system can provide a dynamic support structure for elementary teachers. The SCIED Advisor will not replace the need for extensive in-service training for teachers. It could, however, serve as a structural bridge that supports in service activities. We envision the SCIED Advisor could become a forum where teachers and researchers can share in the dissemination and integration of their work.

References


Scientific American, (October, 1992). Raising the grades in U.S. science education.
Incorporating Guidelines Into A Case-Based Architectural Design Tool

Craig Zimring
Osman Ataman

This paper discusses an ongoing project called Archie, a collaboration between cognitive scientists and researchers in artificial intelligence and architecture, aimed at creating computer-based aids for conceptual design. Archie is a "case-based design aid" (CBDA): a tool that provides designers flexible access to evaluated examples of past experience that they can use in their own designs. Archie is a "clever" hypermedia database aimed at aiding conceptual design in architecture. It contains about 200 problems, responses, stories, and building descriptions derived from evaluations of six libraries and two courthouses.

In this paper we provide a brief history and description of Archie and discuss some issues that have come into focus through developing and initially evaluating the system: how specific architectural case information can be organized; how users can be provided more general information about issues and building types; and how information can be indexed. In each of these we briefly discuss the current state of the system and propose some potential future directions.

Introduction

Architectural design is exceedingly complex and sometimes messy. Designers may start with only a general view of the problem to be solved; an important part of the design process is often to create the problem as well as to solve it. Moreover, decision-making in architectural design operates at different scales in rapid succession and is seldom linear. Designers often move back and forth between strategic and detailed decisions as a way of defining, and redefining, the means, goals, and criteria for evaluating the final design. In fact, a designer may be fairly far along before s/he finally decides which of the client's specifications to satisfy and how the client's goals relate to other aesthetic or cultural interests s/he might have. These complex normative and substantive decisions are difficult to model using computers, and in fact it may not be possible to completely automate design.

Instead, we have spent the past several years using computers to aid rather than automate design. We have developed a series of computational design aids that we call "case based design aids" (CBDAs). CBDAs provide human designers easy and flexible access to past experiences, and allows the humans to make the inferences and to choose how the information will be used in the design.

In this paper we briefly describe a CBDA called "Archie," then discuss three particular issues: 1) How can architectural case information be structured in a database? 2) How can a designer be provided general information in a CBDA while maintaining the advantages of using specific case information? 3) How can cases be indexed for use specifically within architectural design? In the following sections we describe Archie, explore questions of generalization and indexing that developing Archie has raised, and suggest some future directions for further research and development.

themselves over time and adapted themselves to different conditions. In a series of house designs, for instance, Frank Lloyd Wright modified basic models of human habitation to produce a wide range of variations which adapt themselves to specific environments while maintaining their essential organization or logic. Suppose we adopted an organic metaphor, conceiving of communications and
Background and Rationale

Archie is the result of an ongoing collaboration between the College of Computing and the College of Architecture. We are interested in exploring how to develop aiding systems for design, and particularly how access to cases can aid design. We are focusing on conceptual design: during this early stage of design decisions strongly influence the future course of the project and can be changed relatively easily. Whereas there are many kinds of decisions that are important during conceptual design, in Archie we have created a system to help designers understand different intentions for action of a range of stakeholders in a building project, such as users, owners, and builders, and how these intentions relate to design decisions about building form. Archie provides easy and flexible access to case studies of buildings (in the present version, libraries and courthouses), that designers can adapt to their own needs. ARCHIE is intended to help experienced designers make initial conceptual design decisions and/or evaluate their own designs. It also helps novice architects and student users learn design principles, problems, related responses, and lessons.

Archie is based on an artificial intelligence (AI) paradigm called case-based reasoning (CBR) and a specific approach to post-occupancy evaluation (POE) case study evaluation of buildings. Case-based reasoning (Hammond K., 1989; Kolodner J. et al., 1985; Schank R., 1982) is a theory and technology within AI based on the idea that humans often solve problems by using specific past experiences. These experiences, or "cases," are used to explain new situations, are adapted to meet new demands, or are used to evaluate new solutions (Kolodner J. 1993). For instance, if a designer is faced with the problem of designing the information desk at a community library, s/he may remember similar cases s/he has experienced and adapt one or several of these to her current design. Case-based reasoning has been applied to a wide range of domains, such as scheduling (Mark W., 1989), diagnosis (Bareiss E., 1989; Koont P., 1988), planning (Hammond K., 1989), explanation (Kass A. & Leake D., 1988), design (Hinrichs T., 1992; Hinrichs T. & Kolodner J., 1991; Navinchandra D., 1991), and architectural design (Domeshek E. & Kolodner J., 1991; Domeshek E. & Kolodner J., 1992; Goel A., et al., 1991; Pearce M. et al., 1992). Effective case-based reasoning depends on having relevant cases in the system that can be applied to the problem at hand, and on retrieving a reasonable set of relevant cases quickly and easily.

In addition, Archie is based on post-occupancy evaluation (POE). POE is a set of techniques that assesses the effectiveness of buildings that are in use (Zimring C., 1987; Friedman A. et al., 1978; Preiser W. et al., 1987). Although it has been used to examine a wide range of technical and social issues, most POE work has focused on learning how successful a building is for various stakeholders, such as different types of building users, clients, or designers. Our approach to POE focuses on a specific set of design problems: how to resolve multiple intentions simultaneously. Any given design decision often must consider multiple considerations. For instance, in a community library, librarians need to be accessible to patrons, yet be able to get their work done without interruption. Depending on the exact context, a successful design must accommodate both of these intentions. In our POE work, we conduct field case studies of buildings to try to uncover these intentions through observation and interview.

The use of cases in design has multiple advantages. Human experts do not simply use systems of rules, they often access libraries of experiences (Riesbeck C. & Schank R., 1989). Cases are often vivid and specific and encourage the designer to consider how the case fits into the current problem. The effort to adapt cases encourages the designer to develop a mental model of the problem and solution. However, whereas specific cases may carry significant lessons, if the designer does not have a sufficient general framework for understanding them, s/he may not find them useful. In developing Archie, we have linked specific case descriptions of buildings with more general statements.

The structure of content in Archie

Archie is built using an experimental shell intended to ease construction of Case-Based Design Aids (CBDAs) called "Design-MUSE" ("Design
A response should be general, but not too abstract, and should be suggestive rather than imposing a solution on the user. Furthermore, the general response should describe the outcome, which means the resulting state of the world when the solution was carried out. This outcome may need to record points of view of several different stakeholders. Consequently, some possible responses to the problem given above are:

In public buildings, allow an easy access to the restrooms from the outside or entrance hall, after the visitors/users have passed out through check out and control areas.

Place restrooms near: building entrance; supervised children’s reading area, or other spaces where staff members are present.

Locate the entry to the restrooms where users must pass by the desk of a librarian or other staff members.

Stories

A story illustrates a specific example of a general problematic situation. It may also provide a specific example of a general response to a problem. A story consists of the following components:

A brief description of the existing design involving the entities mentioned in the general problematic situation.

Brief interviews of the users involved, verifying the evaluation component. They should mention if the users face any other problems connected with this situation and how it can be solved, and also any suggestions made by users.

Response to the problem indicating if the intentions were satisfied and that mention how and why.

Graphic illustrations to support the story.

Consider the following story that illustrates one implementation of the problem, and responses listed:

In Buckhead Public Library, the public restrooms are located adjacent to the public lobby near the circulation desk at the main entrance. Visitors can easily access the restrooms from the reading areas, but they must pass through the circulation area on the way. However, visitors complain that they have to walk all the way to the entrance.

Nonetheless, this pathway makes it difficult for visitors to smuggle books into the restrooms without being noticed by staff members working at the circulation desk. The arrangement also allows unobtrusive supervision of the visitors without their being subjected to close scrutiny.

The library had been designed so that the public meeting rooms could be kept open after library hours, and the restrooms were accessible from outside the library itself. But now, the access is closed since the public meeting room is not kept open after the library hours. So the only access is from the circulation desk.

ARCHIE’s current implementation on the Design-MUSE involves the retrieval of design "stories" via specific, previously identified architectural "problems" addressed by each of these stories. Each pair of problems and stories is accompanied by appropriate "responses" — how the architect dealt with the "problem" in relation to a particular "story." This format has many beneficial uses for an architect trying to solve a problem she is having with a design she is developing. However, this format does not address all of the needs of an architect who is formulating the design for a new building.

We conducted several small scale usability tests, in which we asked architecture students to use the...
system to evaluate existing plans of small libraries. Whereas students generally found the system interesting and engaging, almost all of them stated that the indexing was confusing and the indexing vocabulary was awkward. They also indicated that they needed more information while viewing the cases' problems and responses. This need was due to factors such as lack of sufficient architectural background, lack of detailed information about the specific case/problem at hand, and, at times, terminological confusion. Consequently, we are currently considering how to add "guidelines" to the system and how to refine the indexing.

Guidelines

The main purpose of guidelines in the Archie system will be to provide general information. This information will also help designers communicate with the system more effectively so that they are able to take full advantage of navigation among building descriptions, stories, problems, and responses.

As they are employed in architecture, guidelines include one or more of several components: (1) goals, such as "make circulation accessible to able-bodied and disabled users"; (2) rules of thumb or standards for implementation and evaluating implementations, such as "all corridors should be 36" clear"; (3) lists of stakeholders and their intentions, such as "disabled people would like to enter buildings in the same graceful way as able-bodied people"; (4) critical design features affecting the intentions, such as "it is critical that the path of disabled people is continuous: that is, it has no steps or barriers anywhere along the route."

There are some points to be stressed. First, we did not intend to cover all related information (and in fact, it is almost impossible to do so). Rather, we hope to help designers understand a set of key issues and give them the background necessary to access the specific information in the system. Second, we believe that there is no single way of approaching the design process. Guidelines in our system should not be seen and used as rigid sets of sequential instructions but rather as loose frameworks. Moreover the guidelines are intended for a wide spectrum of users -- from non-architects to novice architects, from architectural students to experienced architects.

Below is an example of how guidelines may be organized in our system:

Circulation spaces

Entrance

Stakeholders and their activities: The entrance allows people to recognize the way into the building, and should be easily identifiable. Disabled persons, children and older people are frequent users of libraries. They do not like being labeled as different, and if possible the main entry should be easily accessible to everyone. Often people wait outside for the library to open or for a ride and, depending on the climate, need some kind of protection from the weather.

Lobby/entrance hall

Stakeholders and activities: The entrance hall normally accommodates the flow of the public through the space and various other facilities: people looking at the notice board, the building directory, display cases, public telephones. If the entrance hall also serves meeting and/or lecture rooms, etc., consideration must be given to the maximum number of people leaving these areas at any one time.

Several areas may have direct access from the entrance hall, for example, the circulation desk (which should be visible from entrance) and all public areas including vertical circulation which are not behind the control counter; also ancillary facilities such as exhibition space, meeting rooms, cloakroom, lavatories etc. If ancillary facilities are to be used outside library hours it must be possible to close off the library areas, preferably with one set of doors.

Rules of thumb: International Federation of Library Associations and Institutions (IFLA) recommends that 10-15 % of all public areas and 20-25 % of all staff areas should be allowed for circulation. These figures make provision not only for entrance hall, corridors and stairwells

Change and Continuity: A Framework for Invention

"The level of our interaction with computers was based upon our self-images which he separated into four categories: Scientists- application to existing disciplines. Information Specialists- crossing boundaries of disciplines. Entrepreneurs- seeking opportunities for innovation. Students of Change- anticipating new contexts." Russell Aldrich.
but also for cloakrooms and lavatories; the higher figures are for large libraries with a high degree of division into separate rooms/departments.

The guidelines can be used when the user initially enters the system, or at any time to help him/her understand and/or clarify issues.

**An example of using the system**

The following is an example of how the system might be used: An experienced architect is looking into circulation concerns in a library which she is not familiar with. First s/he looks up the Guidelines about "Circulation" to get oriented, as seen in Figure 1.

S/he reads the Guidelines and notices that main entrance access and lobbies are a major concern in public buildings like libraries. As seen in Figure 2, s/he retrieves stories about "Accessibility" in "Public" buildings. By doing that, s/he can use some important keywords related to the Guideline which is provided by the system.

This retrieves stories about lobbies and entrances and general access to spaces from interface spaces. The architect notices that several of the retrieved stories mention noise as a major consideration when organizing the main access to buildings, so s/he decides to retrieve some stories specifically about acoustics.

---

**Figure 1:** The user looks up "Circulation" guidelines to get oriented.

---

**Howard's End**

ACADIA 1994
Library lobbies are more than mere entry spaces. The public lobby is where visitors gather before and after working in the library. Different lobbies serve different functions: some merely channel visitors in and out of the library, others serve as temporary meeting places, information centers, and rest areas.

Some lobbies perform separate social functions, such as serving as notice areas for public events. Consider furnishing bulletin board space, seats, and other furnishings conducive to public interaction.

The Buckhead Library has a large public lobby of approximately 24 sq m. It is large enough to accommodate many patrons at once. The lobby is equipped with bulletin boards, public telephones, waste receptacles, and many other public utilities. The lobby thus serves as an anteroom and a rest area. It opens onto the main, internal library and the Circulation Desk, but double doors between the two sections keep noise from travelling from one area to the other.

Figure 2: The user retrieves stories about "Accessibility" in "Public" buildings.

In Theory of the Avant-Garde, Renato Poggioli points out that many authors view architectural history as movements, periods of consolidation, such as the Beaux Arts, Modern, or Post-modern, interrupted now and then by short bursts of avant-garde fervor and experimentation. Poggioli argues that we can just as easily look at history as long periods of avant-garde activity, with relatively
Problem (1 of 1)

Traffic and Noise in Periodical Section

The Newspaper and Periodical Section is one that receives transitory use. It is often used for meetings, for discussions, and for rest from real work. This can generate high levels of noise.

Response (1 of 1)

Provide Transparent Acoustic Barrier

Provide a transparent barrier between the Newspaper and Periodical room and surrounding areas.

In a closed room, provide a visual opening to one side, or in the door, so that patrons looking for friends can do so without opening the door frequently, and so that staff can monitor behavior.

In Alpharetta Library the Periodical and Newspaper section is located across the catalogues from the Circulation desk. It lies between the Children’s section and the group study tables. Since it is grouped with noisy areas such as the Children’s section, the Circulation desk, and the lobby itself, the noise from patrons using Periodicals and Newspapers is not really obvious to other patrons. At the same time, the staff at the Circulation desk can keep an eye on the patrons using the Periodical and Newspaper section. The main problem from a user’s point of view seems to be the location of the copy machine, which is difficult to locate and placed far from the Periodical and Newspaper section, where it is often in demand.

Figure 3: The user reads all related stories, associated problems and its responses about "Acoustics" in "Public" buildings.
After readings these stories, s/he wants to double check and see if there are any acoustical problems s/he has forgotten about. So, s/he retrieves the Guidelines for “Acoustics”. (Figure 4) As seen in the example, the user has access to detailed information about a particular area throughout the interaction process. Even though some design principles are provided in the Response section, the Guideline section covers the whole investigated area regardless of a particular problem.

For example, the following problem addresses circulation in libraries:

If the public lobby is very small, the visitors waiting for an event in a community room tend to gather inside the library, near the circulation desk, and produce noise that bothers other users.

The responses for this problem are:

1. Provide a separate lobby for the community room.
2. Use stacks or other features to separate the reading areas from the noise of the lobby.

short periods of consolidation or agreement around a given architectural dogma or style. He asserts that the avant-garde, rather than the architectural movements, has provided the central impetus for the development of architecture. After a period of experimentation and rapid change, of great advancements and interesting sidetracks, we appear to be entering a period of consolidation. We're seeing ourselves trying to regroup, to integrate, to bring
The guideline for circulation, however, covers more general principles. In this case, whenever the user gets confused or cannot proceed any further because of the lack of more detailed (or general) knowledge, at any stage, she can access those guidelines in order to learn more about the area.

Indexing

The most important issue in cased-based reasoning is retrieving appropriate cases (Kolodner J., 1993). The index of a single case determines how and when the case should be retrieved. The choice of indexes is very important because the indexes represent an interpretation of a situation, one that takes into account the way the user might think about and the circumstances in which the user might want to recall the situation.

A first concern is what kind of stories are to be dealt with. This system focuses mainly on libraries, but the fact that a library shares many features with other public buildings challenges the indexing vocabulary to be easily scaled to cover other public buildings. In our system, cases about courthouses are indexed similarly with the library stories and show the inter-domain compatibility of the system.

Second, our initial usability testing showed some problems in specific use of indexes by architects. There was a general desire for a different format for accessing the case information. They preferred structures that use architecture's general functional/spatial decomposition as a way of indexing the problems/responses/stories already in the system.

The existing ARCHIE uses descriptive indexing. The indexes are composed of descriptors specifying different design issues, such as structural component, functional system, stakeholder perspective, and lifecycle. Given information about a part of the design, the system searches for descriptions mentioning that part, and finds stories with either "positive" or "negative" outcomes that identify pitfalls and opportunities (Domeshek E. et al., 1992). However, the current indexing style does not have an architectural organization structure. Without having an architectural organization structure, the links between different categories as well as the search process may be confusing for architects.

To resolve this, we are refining the indexing structure as a five-level hierarchy. At the highest level is the core of architectural artifact; the next level is the type of architectural artifact divided into sub-fields (Residential, Non-Residential); at the third level is the type of sub-fields (Public, Commercial, etc.); at the fourth level are the main general spaces used in those sub-fields (Public Space, Interface spaces, Private spaces); and at the lowest level are specific kinds of activities (Lending, Reading, Reference, Seating, etc.). This hierarchy is presented in Figure 5. In addition to representing the structural relations between the five levels, this hierarchy classifies the "fixedness" of the architectural artifact: the spaces in the lowest level can be changed according to the domain (library, courthouse, office building, etc.), while the highest levels remain fixed respectively. For example, the requirements for Circulation of libraries and courthouses are different, while the requirements for Circulation of Public buildings always remain the same.

The main issue is to create an indexing vocabulary general enough to cover the range of tasks the case-based reasoner is responsible for and at the same time specific enough to make necessary differentiations among cases in order to retrieve only a small number of relevant stories for a query.

How the system addresses the needs of the users

The kinds of decisions that users make are related to their design goals and to the information that they need in order to accomplish their goals. A good indexing vocabulary should provide adequate items to handle users' decisions. Initially, our indexing vocabulary used just a functional approach. However, in order to address the users' needs the vocabulary was refined to also use a reminding approach. Consider the Buckhead public library story presented earlier as an example of how the system addresses the design goals.
of the users. Some design goals addressed by this
particular story may be the following:

* designing the position of private spaces (including the restrooms)

* designing the position of the alarm system

* remodeling an old building with a new functionality

In order to evaluate these goals we need the following information:

* statistical information about the books damaged or stolen, in order to know how bad the situation is

* can the presence of storage spaces really improve the situation?

things together. To be effective as researchers, we need to be making some judgements about things that are more effective than others and ways in which we can share the benefits or discoveries of the many diverse programs that have evolved. While this consolidation is important, we need to be careful that it does not cut us off from continued exploration or what we can imagine is possible. If
Here is a check to see how the indexing vocabulary addresses the above users' design goals. The following is an example of indexing these goals in the system:

- **ARTIFACT/Non-Residential/Public/Library**
- **SPACE/Interface-Spaces**
- **SPACE/Interface-Spaces/Restrooms**
- **FUNCTION/Circulation**
- **FUNCTION/Security**
- **STAKEHOLDER/Designer/Architect**
- **STAKEHOLDER/User/LongTime Worker**

However, the remodeling of spaces is not well addressed in the example above. This is not necessarily bad, because the indexing vocabulary does not need to cover everything. The answer to "What may happen if the space is remodeled?" can provide an appropriate indexing vocabulary to cover this aspect. It seems obvious that an improvement is sought. However, it is not obvious that improvements will fail if some fine points are ignored. This aspect of failure may be indexed as:

- **ISSUE/Efficiency or Maintainability**

However, the case is about "non-efficiency", and, therefore, if a user cannot find a particular feature indexed, s/he should try to find an opposite feature as an index and consider only the retrieved failure stories.

**Conclusion**

Constructing the system to be used by novice and experienced users eliminated many of the problems associated with trying to anticipate the expertise of the user. The straightforward approach of the Lesson Window in Design-MUSE gives a clear outline of the information presented so the user cannot get easily lost if they are at least familiar with the Macintosh window format.

The vocabulary, presented as menus, eliminates the problem of users trying to find appropriate terms to retrieve information, although, in general, the user is assumed to be an architect or at least to be familiar enough with the architectural jargon. Further, the implementation of the Guideline dimension provides assistance for filling in a user's knowledge gaps concerning general architectural terms and issues for the novice user and concerning the terms for the Library artifact for an experienced architect out of her domain of expertise.

The functional/spatial breakdown of the indexing vocabulary to represent the stored cases was done specifically with the designer/architect in mind. Spatial entities and functions are the entities that an architect thinks about and addresses when designing. We followed this approach in reorganizing the system: the case presentation first addresses the spatial entities and functions addressed by the particular story. Concluding each story with "user recommendations or comments" is also very "designer logical" as a designer's goal is to satisfy and please the user as much as possible.

In building the initial case library, our approach tries to provide coverage of the common problems encountered in the design of libraries. In the testing and training phase, the inadequacies present in the case library, in the content of cases, and in the indexing scheme should be fixed. It is hard to tell at this point, without significant usability testing, exactly what all of these inadequacies are. However, we discovered up front that, because each story addresses so many issues in a design or could help in so many different design situations, many of the indexes for stories are similar. This often leads to the retrieval of large numbers of stories that the designer has to wade through. In order to retrieve a limited number of the stored stories, using the current search option, "Matching Any of the Interests" choice should be avoided, while using, "Matching Most of the Interests" seems to work well. Along this same line, the more specific the user is up front about his/her area of interest the better. If no suitable stories are found, one can make retrieval specifications gradually less specific to enlarge the number of stories or problems s/he retrieves to browse.

**ARCHIE currently exists as a set of 200** analyzed stories, 82 problems, and 148 responses, plus building descriptions, derived from evaluations of six libraries and two courthouses. We are refining the
indexing to provide user support for a broad set of stories and problems. We are adding more stories to more generously fill out the indexing hierarchy. The most significant addition that should be made to this system to make it more useful is the inclusion of complete architectural drawings, corresponding to each story, that could be brought up in the Design window when a story is retrieved. This visual information is crucial if this system is to provide significant help to an architectural designer. Designers are graphics-oriented. Shown below is a sample run from current ARCHIE (Figure 6).

Figure 6: Sample run of ARCHIE in its current state.

---

we cease to wonder what might be, there will cease to be anything new. We need to provide a context for continuity and change, for communication and invention.
As a general case-based reasoning tool, this application of Design-MUSE takes advantage of one of the most significant offerings of CBR: extended, non-volatile memory. As a reminding or educating device, the system's ability to store and retrieve far more information than any architect, experienced or novice, can remember is its key attribute. Our goal is to support the creative development of an architectural design idea.

We are currently initiating a program of testing Archie with architecture students at various levels, from beginning to advanced. We are also revising the interface to make the system more intuitive to use, and are adding new material to the system.

Acknowledgments

The Archie project has involved many people and benefited from the contributions of many others not directly involved. In addition to Janet Kolodner and Eric Domeshek, who initiated this project, and Ashok Goel and Richard Billington, who helped shape this project at the earliest stages, we have been fortunate to work with CS students Anna Zacherl, Vijaya Narayanan, and Marin Simina, architecture students Ali Malkawi, Ellen Do, Amin Farooq, and Hussam Khalil. This work has been supported in part by the Defense Advanced Research Projects Agency, monitored by ONR under contract N00014-91-J-4092. All views expressed are those of the authors.

References


Howard's End


"Invention can only be done deliberately if the inventor can discern similarities between the particular result which he is envisaging and some other actual result which he has seen and stored in his memory. ... An inventor’s power to invent depends on his ability to see analogies between results and, secondarily, on his ability to see them between devices..." David Pye
THE DESIGN OF A TOOL KIT FOR CASE-BASED DESIGN AIDS

E. A. DOMESHEK, J. L. KOLODNER
College of Computing, Georgia Institute of Technology
Atlanta GA 30332-0280 USA

AND

C. M. ZIMRING
College of Architecture, Georgia Institute of Technology
Atlanta GA 30332-0155 USA

Abstract. This paper extends the discussion of Case-Based Design Aids (CBDAs) presented at the 1992 AI in Design Conference (Domeshk and Kolodner, 1992). After reviewing the CBDA concept and its motivation, we discuss progress in implementing such systems, and focus on the generalization of our original CBDA (first developed to support architects with the conceptual design of buildings) into a tool kit applicable to a wide range of design domains. Experience in gathering and organizing case materials, and consideration of more use-scenarios have led us to reorganize and refine many aspects of the original proposal. Here we report our experience building CBDAs for architecture and for jet aircraft subsystem design using the CBDA tool kit Design-MUSE.

1. Case-Based Design Aids: Background and Review

At the 1992 AI and Design Conference we presented a paper describing early work on a case-based design aid for architecture (CBDA) (Domeshk and Kolodner, 1992). That paper focused on two ideas: the importance of conceptual design, and the usefulness of past experiences during conceptual design. Conceptual design is the very earliest stage of design, during which the main tasks are analyzing the problem and forming initial commitments towards a solution. We argued that providing easy access to descriptions and evaluations of previous designs (e.g., existing buildings, with their strong and weak points) would be a major aid in those tasks. In this paper, we briefly review this argument, and then discuss our progress, emphasizing efforts to generalize our original conception and the tools intended to support it.

1.1. CBDAS: THE MOTIVATION

We conceived of CBDAs as a way to apply insights and techniques developed in the AI paradigm of Case-Based Reasoning (CBR) (Kolodner, 1993) so as to have a real effect on the quality of design processes. Following on several
interesting, but limited, autonomous design systems (Goel, 1989; Hinrichs, 1992) we decided to redirect our efforts towards building *tools* intended to *aid* people doing design. By adopting this strategy we expected to be able both to demonstrate a useful application of CBR and to advance certain aspects of CBR and design research. In particular, we want to learn how to segment and index large complex cases in domains where no clear causal models are available. We also see CBDA research as a vehicle for exploring how domain knowledge can be structured to aid conceptual design.

Our CBDA research has been driven by, and has aided, an ongoing exploration of conceptual design. Conceptual design caught our interest, as an under-studied aspect of design, much in need of methodological analysis and automated tool support: on the one hand, relative to the time and effort it consumes, conceptual design has a disproportionate impact on the ultimate cost and quality of a designed artifact; on the other hand, as a loose informal process, conceptual design is little studied in computer-oriented circles.

The very informality of conceptual design is an argument for CBR's relevance to the task. At its core, CBR claims that the basis of expertise is experience, and that the first line of attack on any new problem is to seek applicable lessons in old situations. CBR is a particularly appropriate strategy when problems are open-ended and when a reasoner lacks a strong domain theory to support rule-based analysis. The notion that CBR is relevant to conceptual design gains further support when you look to the real world and observe design practitioners making significant use of previous cases as they work through the early stages of their problems.

Cases can help designers in several ways. Prior experiences considered during conceptual design can suggest approaches to solving a problem or warn against proposed solutions that have been tried and found wanting. Just as likely, however, cases can serve primarily to elaborate the problem statement itself, illustrating sometimes non-obvious issues that deserve consideration, and offering ways of evaluating and critiquing partial solution proposals whatever their source.

1.2. CBDAS: THE CONCEPT

The cognitive model underlying CBR and observation of designers at work on concept development both suggest that conceptual design would benefit from a quick, easy way to survey a wide range of existing artifacts. Currently such surveys are time consuming, costly, and prone to omissions. Architects, for instance, often flip through magazines and journals, but these are not particularly well indexed, nor do they contain much analytic material; less frequently, architects will travel around the world to visit relevant or landmark buildings as they research new commissions. With options like these, it is no
wonder that the research process often misses many relevant precedents; still less wonder that it will almost always miss interesting lessons that could be learned from less obviously relevant buildings.

One way to think about a CBDA is as an on-line library of design experiences that saves the designer from having to trek to the library (or around the world), and that does a better job of collecting, organizing and presenting experience-based design lessons when they are relevant. Our previous paper dwelt on some consequences of this conception, most notably the need to combine raw documentation of existing designs with evaluations of the resulting artifacts, the need to carve the evaluative discussions up into small chunks focused to teach lessons about particular design issues and their interactions (yielding presentations we call stories (Schank, 1991)), and the need to index and cross link the resulting stories with the design documentation, with design guidelines, and with other stories as well.

A CBDA is a clever hypermedia system. The cleverness lies in the kinds of information included, the ways that information is presented, and the kinds of connections, or forms of browsing, supported. Our strategy is to understand the sorts of questions and issues designers should be considering, what they need to know in order to arrive at decisions, and how they prefer to visualize and organize the required knowledge.

Figure 1, for instance, shows a piece of a screen shot from our most recent CBDA, MIDAS1 (Domeshek, Herndon, Bennett, and Kolodner, 1994), displaying a graphic of the hydraulic system from the A-7E jet. The light and dark colored dots on the schematic with lines leading off to annotations in the left margin are indicators of positive and negative stories associated with the hydraulic system of the A-7E. For instance, clicking the mouse on the annotation, “Using MS-6 resulted in increased component size and system weight” would lead to the display of the following story:

MS-6 is being evaluated as a non-flammable replacement for MIL-H-5606 and MIL-H-83282. When compared with these fluids, MS-6 has high viscosity, high density, and low bulk modulus. In an A-7E high pressure application study, MS-6 was substituted for MIL-H-5606 and tested for its performance.

The system internal leakage was extremely low, but the fluid did have the tendency to foam and absorb air. This caused poor pump performance and resulted in a redesign to increase line diameters to maintain the system pressure drops. Since the total system volume increased, the reservoir size had to be increased as well. These redesigns for the MS-6 fluid resulted in a weight increase of 11.8% and a volume increase of 6.3%.

---

1MIDAS stands for: Memory for Initial Design of Aircraft Subsystems.
Stories like this do not stand on their own as isolated, uninterpreted experiences. A CBDA makes sure to connect them to other stories that touch on related topics. One way this is done is by linking stories to statements of the problems they exemplify or address. The following is a problem statement covering the MS-6 story:

While traditional hydraulic fluids provide effective power distribution at a reasonable cost and weight, they tend to be highly flammable. Hydraulic system fires have caused much loss of life and introduced significant extra costs.

Stories are also linked to statements describing possible general responses to recurring problems. Here is a response to the problem above that also relates to the MS-6 story:

Non-flammable or flame-resistant hydraulic fluids should be used, where feasible, to reduce the chances of a system failure caused by a fire. Non-flammable fluids currently under development are, however, both heavy and costly. Overall system trades are necessary to determine the applicability to each design case.

None of these texts is as effective standing on its own as when appropriately linked to other presentations. CBDAAs provide a framework for useful information and for the connections that enhance each item's value.

Given the right information and a set of meaningful connections, the final piece required to make these systems work is an indexing scheme that assures easy initial access to relevant materials. We assume users will be engaged in
working out designs, so our indexing scheme associates stories and problems with the issues they address. Indexes should allow users to describe their current commitments and pending decisions so as to retrieve relevant advice.

1.3. CBDAS: THE HISTORY

The longest running CBDA project has produced a series of variant systems going under the general name of Archie. The extended project, however, has been through several distinct conceptual and implementation phases. The first attempt, described in (Pearce, Goel, Kolodner, Zimring, Sentosa, and Billington, 1992) established the centrality of case data and the focus on conceptual design issues. However, the system produced was inadequate as an aid for designers on a number of practical grounds, most notably its monolithic representation of design cases, and its inability to communicate design lessons to users (Domeshek and Kolodner, 1991). One response was presented in our AI in Design '92 paper (Domeshek and Kolodner, 1992), which described our plan for a story-centered hypermedia browsing system organized around artifacts and design issues. At that time, we had only a SuperCard mock-up, serving primarily as a tool for interface experiments.

By the fall of 1992 we had a more realistic implementation of Archie-II running on Symbolics Lisp Machines (Domeshek and Kolodner, 1993). This prototype supported issue-based search and a case library of over 100 stories. It allowed us to begin experimenting with memory organization and indexing (Zacherl and Domeshek, 1993). Unfortunately, this prototype suffered severe performance, reliability, portability, and generalizability problems. Therefore, in March, of 1993, we began to completely rebuild the system, this time as a more generic tool kit hosted on a Macintosh computer.

The move towards generality in this re-implementation was, in part, driven by the MIDAS project, then just getting under way as a joint effort with engineers from Lockheed Aeronautical Systems Company. MIDAS was to be an Archie-like system supporting early design of aircraft subsystems; an important additional goal was to build the system so that it could largely be constructed and maintained by domain experts (rather than by AI experts). A first version of MIDAS was successfully prototyped on top of the newly built CBDA tool kit in four months. The screen shots and data in this paper are drawn from MIDAS. Meanwhile, collection, analysis, and preparation of data for Archie-II has ramped up, so we now have over 200 stories, drawn from four courthouses and six libraries. We are evaluating Archie-II in the context of an undergraduate architectural design studio this spring.

2An alternate response was developed by other members of the original Archie team, leading to a system called ArchieTutor (Goel, Malkawi, Pearce, and Liu, 1993). That project is no longer under active development.
2. Design-MUSE: A CBDA Tool Kit

The CBDA tool-kit, Design-MUSE, was designed based on our experiences building Archie-II and MIDAS, and both have been reimplemented using Design-MUSE. Systems built in Design-MUSE support three different classes of user: 1) end users who simply browse through the available materials; 2) expert users authorized to augment the materials in the library; and 3) system administrators authorized to redefine data structures, including the indexing framework and its vocabulary. These three levels of privileges are called browsing, modifying, and defining. Whenever possible, the system presents a consistent interface no matter which of these three modes it is in.

2.1. CBDA SCREEN LAYOUT

Figure 2 sketches a CBDA's structure as seen by a browser. The interface has four windows, each managing a major class of information. The system entry point and the intended conceptual center is the "Notebook" window. Since the point of using a CBDA is to advance an ongoing design project, the centerpiece of a CBDA should be a personalized notebook in which a designer records what they have learned and decided during their browsing. The two large windows beneath the "Notebook" form the system's core.

The "Designs" window handles documentation about particular artifacts. It has three major segments. The Display area offers basic documentation on a design; currently it presents only graphics, such as the hydraulic system schematic in Figure 1, but it will soon be extended to handle textual and tabular data. Coupled with the display, the Annotations area presents short (usually one sentence) summaries of available stories discussing lessons learned from the artifact. Finally, the Descriptions pane contains a form so users can choose features to characterize artifacts (e.g., buildings or aircraft).

The "Lessons" window organizes and presents the evaluative materials that allow users to learn interesting lessons from artifacts. It is composed of four panes. The Interests pane contains a form allowing the user to specify a set of design issues and artifact pieces that together characterize a focus for design exploration. The Problems pane discusses general negative outcomes observed to impinge on the specified issues. The Responses pane collects general ways of ameliorating those problems attempted in previous designs. The Stories pane presents evaluated descriptions of particular designs that either illustrate the problem, or the success or failure of some response.

3 In the current implementation, the notebook is only a shell of its intended function: it mimics the look of a notebook and provides a way for users to log in (thereby setting the system's mode according to their assigned privileges), but it does not yet allow for the accumulation and annotation of browsing results.
Figure 2. Schematic of CBDA browsing windows.

An important adjunct to both of these windows is the "Sources" window where all information in the system can be tied to citations. This is important both to give credit to those who contribute their knowledge to the system, and to allow users some way of deciding how much to trust the information presented to them. Much of the information in the system—particularly the evaluations—is informed opinion: the (annotated) citations in the sources window turn stories into something like signed editorials.

The Designs, Lessons, and Sources windows share a common structure in that they are composed from a set of resizable panes, each dedicated to displaying a particular type of information. While a pane can display only one item at a time (e.g., a single story or a design graphic), it might be assigned some selected set of items drawn from the class of objects it is designed to manage. Thus there may be selected sets of designs, problems, responses, and stories. Figure 3, for example, shows the Stories pane from the MIDAS system, at a time when it is displaying the second story from the current set of eight selected stories, all chosen for consideration when the user specified an interest in safety and hydraulic system fluids.4

The system automatically keeps its several information presentation panes synchronized. For example, when the user chooses a story, the system makes sure that the Problems pane contains discussion of those problems reflected in the story, that the Responses pane contains any relevant responses, and that the Designs pane contains documentation about the artifact being discussed.

---

4Our domain expert considered stories about electrical systems designed to replace hydraulic systems relevant when considering requirements (such as safety) that were issues for both design approaches.
in the story. At any time, one of the lesson panes—Problems, Responses, or Stories—may be the focal pane. The focal pane is the one whose displayed item was chosen by the user and which controls what is being displayed in the other panes. The focal pane is highlighted so it is easy to keep track of the relationships among the information in the several panes.

2.2. BROWSING A CBDA

There are effectively four ways to search and browse in a CBDA. Interest-based search uses the system’s issue-centered indexes to find lessons learned (problems, responses, and stories) that match a user’s concerns. Problem-response-story browsing allows the user to move through the network of lesson components so that concrete examples are set in the context of design decisions, while abstract statements are illustrated by the specific examples. Artifact-centered browsing lets users quickly scan lessons directly associated with existing artifacts they have chosen as relevant models for their current design. Description-based search finds artifacts using indexes that describe design features (in contrast to the design issues used for lesson retrieval).

2.2.1. Interest-Based Search for Lessons

In interest-based search, users build search cues in the menu-driven, fill-in-the-blank Interests pane. To retrieve lessons that support designers’ decision
making, interests must include a specification of which parts of an artifact are being worked on and what issues are currently of concern. Significant lessons usually report interactions among issues, often affecting several parts; they represent tradeoffs that past designers have made (knowingly or not). Interests therefore allow the juxtaposition of multiple parts and issues.

There are several sorts of parts. In particular, for the physical artifacts we have so far considered, we focus on physical parts and functional parts. In building design, for example, physical parts translate to spaces (such as lobbies and courtrooms), while functional parts are systems (such as plumbing and electrical wiring). In aircraft subsystem design, we are mainly concerned with functional systems, such as the hydraulic system, and their component parts, such as pumps, lines, and fluid (see Figure 4, for example).

By issues, we mean to include the broad range of things that designers must decide upon and that their decisions affect; this includes 1) explicit and implicit goals (the requirements or intents of the design), 2) the flip-side of such intents, properties to be avoided (the pitfalls that may yield an unacceptable design), and finally, 3) the specification of particular features, components or parameters as part of the artifact design.

In our current systems, we have attempted to keep the definition of issues broad by explicitly including in the interests specification a way of indicating the relevance of multiple stakeholder viewpoints and the different phases of the designed artifact’s life cycle. This fits well with modern methodologies like concurrent engineering, and accounts for Lockheed’s involvement with these design aids. Figure 4 presents a sample interest for the MIDAS system.

2.2.2. Problem-Response-Story Network Browsing
The second browsing mechanism assumes that some problems, responses, or stories have already been selected (possibly through the interest-based search just described). Browsing proceeds from that point through the network of links among these presentations. Problems, responses and stories should

---

5 In other domains, dealing with different kinds of designs, we can imagine picking out temporal or perhaps even social parts, as in a process or an organization.
Figure 5. Problem, Response, and Story related to Story in Figure 3.

together provide specific and memorable advice, while setting the advice in a more general context. The general context, in turn, is made clearer by the specific stories that flesh out what might otherwise be vague abstractions.

A story describes some aspects of a design and recounts consequences for some of the resulting artifact's stakeholders. Figure 3's story, for instance, recounts a failure in routing electric control lines. If that story is of interest to a designer, then chances are there are other stories in the system that would also be of interest—stories of other problems encountered in line-routing, stories of successful instances of routing lines, and so on. There are likely also problem and response statements in the system relevant to the designer's interests. For instance, one problem discusses interactions between efficiency of wire runs and safety of the routing system, that problem in turn provides access to other stories, such as a case where flight surfaces in a test version of the C-141 moved without pilot commands because of electrical interference. Figure 5 shows this set of texts in the Problems, Responses, and Stories panes.

The lessons network is traversed using the "Browse" menu in the header of most panes (see, for example, Figure 5). In the Stories pane, that menu offers ways to see related problems and responses (as well as design graphics and source citations). Browse menu commands gather linked items and load them as the new selected sets in their panes. From the Responses pane, for instance, one option picks stories that illustrate successes in implementing the response, another stories that illustrate failures of the response. In both cases, browsing leads to selection of new stories in the Stories pane. From there, of course, it is possible to browse to other problems or responses.

2.2.3. Artifact-Centered Browsing

Interest-centered search and browsing through the network of problems, responses and stories makes sense when designers are engaged in their task to the point where they have identified a set of concerns. Before a designer can state a coherent set of concerns, however, it is often useful to browse existing
artifacts to discover the issues that ought to be in play. Artifacts can also organize retrieved stories when a set of interests has been identified. Finally, after studying the issues, it may be useful to return to the bigger picture and examine the solutions of the past without straining them through the sieve of a specified set of issues. Artifact-centered browsing supports all these needs.

A user can start from a particular existing design, and the CBDA will show documentation in the Design window accompanied by indications of relevant stories. In the prototypical case, the documentation is a building floor plan, or an aircraft system schematic, and a story indicator is a small dot placed next to an implicated part of the diagram. Selecting a set of dots causes their annotations to be displayed along-side the graphic as in Figure 1 (and loads the associated stories as the selected set in the Stories pane). Annotations are one sentence summaries of the associated stories. Clicking on an annotation takes the user to the Lessons window, where the associated story is displayed. This then provides an entree to the problems-responses-stories network. Since it is always possible to copy the description of a displayed story to the Interests pane, it also offers a way to begin a new interest-based search.

When a search results in retrieval of many stories, the user can browse the retrieved design documents corresponding to those stories. The documents are accompanied by annotations, as in Figure 1 (showing results of a search for stories about the interaction of hydraulic system pressure and weight). The annotations provide a quick way to survey and organize the results of the search. As before, the user may select any visible dots to shift from exploring the original interests to browsing for other interesting issues.

Finally, browsing through the documentation for an existing design can be informative even without constant detours to explore associated stories. This is particularly true once initial exploration has sensitized the designer to the relevant issues. The designer should then be able to do much of his or her own critiquing, and should be on the lookout for solution features that address those issues that are most salient in the current situation.

### 2.2.4. Description-Based Search for Artifacts

In many cases, the most likely starting point for a CBDA browsing session is a set of precedent designs. What we want is something similar to interest-based search, but to retrieve artifacts rather than lessons. The Descriptions pane of the Artifacts window provides this ability, playing a role much like the Interests pane of the Lessons window. The user fills in a partial specification of an artifact, and the CBDA searches for similar artifacts.

This final form of browsing resembles a tool being built at Lockheed: that system has as its main purpose the identification of the most similar existing aircraft to a new concept under development. Based on features such as mission, range, payload, etc., the tool tries to retrieve documentation
on the most relevant aircraft in its database to support sizing estimates for utility subsystems. This kind of gross artifact-level matching is often a reasonable way to get started on a new project. Our claim, however, is that it is only a way to start, not a full answer in itself. In our CBDAs, we see much of the utility of this type of search deriving from the resulting access to more specific stories, with the concomitant raising of issues, highlighting of interactions, and proposal and illustration of responses.

2.3. MODIFYING A CBDA

Design-MUSE offers major benefits over earlier CBDA implementations: It allows domain experts to handle many aspects of building and maintaining a system. This was one of the major goals of the MIDAS project. A corporate memory stands a better chance of catching on and of being sustainable if its users control what goes in, and if it keeps administrative costs low. We split system-building chores into two sets: modifying, and defining. Modifying covers simple tasks that can be mapped more directly to the normal browsing interface. Defining requires more knowledge about system internals.

The major difference between the system as it presents itself to a browsing user and the system as it presents itself to a user with modify privileges is the addition of a new "Modify" menu to most panes (this menu is visible in all the panes in Figure 5). In addition, where Browse menu items normally traverse existing links (picking out new selected sets for other panes), in modify mode, a user can initiate the creation of new links simply by holding down the option key while selecting an item from the Browse menu.

Modify menus typically offers three major sets of commands: 1) creating, copying, and deleting a pane's items; 2) selecting all available items for a pane; 3) inspecting and specifying hidden background information about a currently displayed item. In the Story pane for instance, the first three items are "New Story", "Copy Story", and "Delete Story". Adding a story to the system starts with the "New Story" item, which creates and displays a blank story. When the user has modify privileges, the title and text fields of the Story pane allow text editing. A title and text can be typed, or they can be cut and pasted from elsewhere (including other application programs).

There is, however, more to a story than the display in the Story pane. The last two items on the Story pane Modify menu, "Story Fields..." and "Story Interests..." bring up dialogs that allow users to fill in additional information (such as story summaries, and PICT files for illustrations) and assign indexes (using a display just like the Interests pane). Finally, the user can create links among presentations by selecting the desired kind of link from the Browse menu while holding down the option key, maneuvering in the target pane to the desired item, and clicking "Finish Link" on the floating dialog that
appeared when the Browse menu item was chosen. A similar mechanism creates links that appear as dots on design graphics. The linking dialog can be used to delete existing links as well as to create new ones.

2.4. DEFINING A CBDA

Users with define privileges have access to one additional window. This “Define” window is composed of three panes. The *Fillers* pane allows for the definition of the menu items presented to the user as options for filling out the Interests and Descriptions forms. The *Classes* pane allows for the definition of the forms that appear in the Interests and Descriptions panes. The *Objects* pane allows for the creation and editing of any type of data object in the system. Figure 6 shows the Define window displaying a collection of MIDAS’s internal data structures.

Some facilities of the Define window are not yet fully operational. While filler editing is quite robust, and object editing works reliably, class editing is still rudimentary. Work on these gaps is in progress. Beyond the definition and instantiation of data structures supported by the Define window, there is just a very small amount of custom code required to create a new CBDA; even this could with some effort be handled in a more user-friendly way.
3. Discussion

In this section, we consider the implications of Design-MUSE for our ongoing CBDA research program. The tool kit was a major programming effort. In the future, we expect three types of return on this investment: 1) For any particular CBDA, the tools in the Define window offer us freedom to experiment with index representations; 2) Our ability to quickly create new CBDA should help us spread and test our ideas about aiding tools, and also develop more general theories of design indexing; 3) Overall improvements in the look, feel, and usability of the CBDA should improve their appeal and effectiveness. But the tool kit's impact is not limited to future experiments; it's evolution spurred, and was shaped by, ongoing study of design theory and designers' needs. The differences between the structure of the current tool kit and our earlier proposals reflect what we have learned along the way.

3.1. THEORETICAL ANALYSIS

From original conception to current implementation, the most important changes have been in our ideas about who our users might be, what CBDA might do for them, and how, in consequence, a CBDA should be organized. From our initial focus on the working designer, we have become concerned also with student designers, and with the many varied stakeholders forming an extended design team. We will discuss briefly some of the constraints that derive from considering these different user groups and use scenarios.

When aiming to help practicing designers, two issues are paramount: First, what would constitute an interesting or useful lesson, and second, what would pass as an acceptable and useful way to communicate such lessons? Many of the stories in earlier versions of Archie were what we called "single point" stories -- stories that addressed a failure or success with respect to one issue. It became clear that practicing designers would, for the most part, find such straightforward stories uninteresting: single point stories often come across as simple reminders (or nagging) about the obvious. The interesting stories were "interaction" stories—stories analyzing tradeoffs in existing designs. Interaction stories should be more interesting to the professional, particularly when the stories emphasize feedback from actual stakeholders. The second point about effective communication is related: in earlier CBDA, the generalizations now presented as problems and responses were lumped as guidelines, and we had difficulty wording these so that they did not sound like attempts to prescribe simplistic rules of good design. We think that breaking guidelines into problem statements linked to sets of response suggestions works better because the system now embeds the assumption that there are usually many reasonable ways to resolve any given design situation.
Splitting problems from responses was also motivated by concern for the needs of student designers. Effective design requires a deep understanding of the problem (that is, knowledge of what you should be trying to achieve), and of the possibilities for solution (that is, technical knowledge and facility, combined with creativity). Experience with architecture students suggests that some get wrapped up in problem analysis, while others revel in proposing interesting forms; but students with either bias are frequently unable to make the connection with the other half of the design process. One of the goals for our CBDAs, then, is to make explicit the connections between the issues that need to be addressed (as expressed in interests and reflected in problems) and the design moves that can be used to address them (presented as general responses, and illustrated through specific stories).

CBDAs’ orientation towards concrete examples looks like it will also pay off in situations where design requires collaboration among diverse members of a large design team. These “stakeholders”, with their varied backgrounds and knowledge, represent different points of view. Communication should improve when all stakeholders can refer to concrete instances rather than to abstract categories that everyone is free to imagine differently. Bringing specific stories that emphasize the concerns of different stakeholders into the conversation should help participants confront a range of issues that might not be fully represented otherwise. It is a rare building design team, for instance, that includes members of the maintenance staff, or in which such representatives feel free to speak candidly and are listened to seriously.

3.2. RELATED WORK

There is notable related work on design aids that capture design experiences or record information about design problems and responses. Probably the closest research is that of Gerhard Fischer’s group on design environments (Fischer, Lemke, McCall, and March 1991; Fischer and Nakakoji, 1993). They have built a series of environments, such as the Janus system for architecture, that integrate a range of support for design tasks. In addition to specification and construction components that allow users to create new designs, a rule-based critiquer of resulting constructions, and some artifact simulation, their environment also includes a searchable “catalog” of prior designs from which pieces can be borrowed, and an argumentation hypertext discussing rationale for various design decisions. We want to contrast our work with their notion of what constitutes a design case and what sort of information should inform design decisions.

In Janus, a prior case is complete design of some artifact. Janus supports kitchen design, so a case is a complete kitchen recorded as an arrangement of kitchen components such as a sink, a stove, a refrigerator, walls, doors, and
windows. In our CBDAs, a case is a collection of design documents (such as requirements documents, floor plans, or system schematics) plus a collection of lesson-bearing stories about the resulting artifact. With respect to a design proper, the information we can collect is weaker than theirs because our systems do not include design construction components; we are, for now, limited to accumulating documents created using other tools. On the other hand, this limitation reflects our assumption that conceptual design requires more than configuration of given components; we are interested in studying the variety of design documents produced during a project.

With respect to design lessons, we believe our CBDAs are unique. Rather than cribbing pieces of a final configuration, we expect reuse from previous designs to focus on appreciation of the consequences of decisions as viewed by the full range of stakeholders over the full life cycle. This kind of evaluative feedback from existing artifacts is not possible if a design aiding system is closed within the world of the designer. A system like Janus can easily accumulate new cases because every new construction can be stored away to be retrieved in the future and borrowed from. But without feedback from outside the design process, there is no real way to learn whether what was constructed (and what is about to be borrowed) was actually a good idea.

In Fischer’s systems, critics and argumentation hypertext offer feedback from the perspective of earlier designers. The argumentation component, intended to record and justify design decisions, is a kind of issue-based information system (Kunz and Rittel, 1970) related to systems like gBIS (Conklin and Begeman, 1989). These systems share an orientation towards capture and representation of design rationale—records of what designers considered, what they decided, and why they decided that way. In some ways, our network of interests, problems, responses and stories is similar to rationale records. The differences include our notion that the arguments should be tied to the experiences of actual stakeholders with existing artifacts, that all decisions on any given project are not of equal interest or importance, and that it may make more sense to help designers fuse the wisdom of experience than to try forcing them to invent and record detailed rationales.

Others in the AI and design community have considered different ways in which existing designs can help new design efforts. Closest in spirit to our CBDA work is the proposal by Oxman (1993) notable for the observation that buildings do not all have lots of interesting lessons to teach; she proposes focusing on precedent buildings—especially important or innovative designs. Most other related efforts differ from CBDAs in placing much more of the design burden on the computer. That goal requires explicit representation of parametric or causal knowledge, to adapt old cases to new circumstances (Goel, 1989; Sycara, Navin Chandra, Gutjahr, Koning and Narasimhan, 1991; Hua, Smith, Faltings, Shih and Schmitt, 1992; Borner, 1995).
3.3. OPEN ISSUES

The fact that our CBDA systems are currently information exploration tools rather than design creation tools reflects our initial emphasis and current resources, not our ultimate intentions. We believe that users will get the most out of a CBDA only when rich information resources are tied to a task and a design context in which some particular information becomes relevant. We hope, over time, to more closely approximate integrated design environments represented like Janus, but we intend to keep our focus on conceptual design of complex artifacts. We will invest more to provide a variety of construction views, and less to support rule-based analysis of what is constructed.

Sticking more closely to our current work, there are important practical and theoretical issues yet to be tackled. In particular, the development of Design-MUSE now puts us in a position to confront problems of scaling up and of generalizing these systems. Putting together a well stocked CBDA is at least as much work as producing a large, well illustrated, coherent book. We must improve our ability to handle large volumes of data, including revisiting interface decisions as changes in the data quantity result in changes in interaction quality; as our libraries grow, we must track new requirements on the content of indexes and the processes that manipulate them.

We want Design-MUSE to become a generic CBDA tool kit that offers strong guidance on system construction—that helps domain experts prepare, link, and index lessons. This requires us to build a wide range of prototype CBDA systems to study their similarities and differences, to note and eliminate those aspects that give users trouble. We can make preliminary statements about how to analyze a domain, how to write problems, responses, and stories, and what the outline of a useful indexing system would look like. Refining and deepening our understanding of generalized design lesson indexes is the most serious of the outstanding challenges. It is here, that our CBDA work will make most its most significant contact with Artificial Intelligence literature and research.

Acknowledgments

Thank you to Ashok Goel, Richard Billington, Anna Zacherl, Vijaya Narayanan, Marin Simina, Ellen Do, Hassum Khalil, Ali Malkawi, Ameen Faruck, Osman Otaman, Marcia Herndon, and Andrew Bennett who have all contributed to or influenced Archie and MIDAS. This work has been supported in part by the Defense Advanced Research Projects Agency, monitored by ONR under contract N00014-91-J-1092, and by the Lockheed Aeronautical Systems Company. All views expressed are those of the authors.
References


USING DIAGRAMS TO ACCESS A CASE BASE OF ARCHITECTURAL DESIGNS

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

AND

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. The Archie case base 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 base 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.

In this project, we are linking two tools—a computer-as-cocktail-napkin program and a multi-media case-based design aid—as an effort to provide an improved integrated tool for conceptual design and as a platform for exploring a range of significant issues. As a tool, we believe that this will be a natural way to retrieve cases, adapt cases, and keep notes about cases. It has been a longstanding concern that designers have to interrupt their normal design process to use a case-based design aid. In addition, this project allows us to study a number of interesting questions: Are there common glyphs or sketching conventions that can be incorporated into a system such as this? Is there a finite range of issues and relationships that can be recognized in sketches or is sketching entirely individual and ideosyncratic? How can graphic-based indexing be incorporated into a case-based design aid? How can loose freehand sketches be linked to more complex and detailed designs? How can the computer aid the progression from the normal quick sketches of preliminary conceptual design to the hard-edged and more precise world that is the normal strength of CAD?

In this paper we provide some preliminary answers to these questions and discuss a path for further research.

1.1. 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 and evaluations. 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 and 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 development, 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, 1994). 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.

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
Using Diagrams to Access a Case Base

Recognizing hand-drawn input (Rubine, 1991; Goldberg and Richardson, 1993; Zhao, 1993) and the use of shared drawing surfaces in design (Tang, 1989; Minneman and Bly, 1991; Bly, Harrison et al., 1993; Harrison, 1993). A recent AAAI symposium on diagrammatic reasoning and representations explored a diverse range of topics (Chandrasekaran, Narayan et al., 1993); several specific efforts are investigating the role of diagrams in reasoning in various domains, including physics and geometry problem-solving (Novak and Bulko, 1990; Koedinger, 1992; McDougal and Hammond, 1993). Others have examined the role of drawing and sketching in design (Lakin, Wambaugh et al., 1989; Ullman, Wood et al., 1989; Faltings, 1991; Goldschmidt, 1991; Ferguson, 1992; Goel, 1992). Several efforts aim to recognize and parse diagrams as expressions in "visual languages" (Wittenburg and Weitzman, 1990; Golin, 1991; Helm, Marriott et al., 1991; Futrelle, Kakadiaris et al., 1992). Finally, the semi-automatic production of design diagrams by computer programs has been explored by Dave (1993) and Ervin (1989).

2.1. 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. Because the bounding box is constructed from the input glyph, recognition is independent of the aspect ratio. 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-url)
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.

![Diagram]

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

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.

2.2. 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 of the connections between a courtroom, a buffer zone, and a waiting area. The program has identified the sequences of letters as words, and the bubbles containing words as diagram elements named ‘court’, ‘buffer’, and ‘wait’.

![Diagram of courtroom connections](image)

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

### 2.3 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:

```
(type-of g1 line)
(type-of g2 circle)
(type-of g3 circle)
(-connects g1 g2)
(-connects g1 g3)
(above g2 g3)
```

*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.

2.4. 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 aid (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 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 illustrative drawings) associated with each text item is scanned and manipulated so that it will appear clear on the screen.

1 We refer to it simply as Archie.
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). 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 need to 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".
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.

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](image)

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

![Diagram](image)

**Figure 8.** Illustration and diagram: "benches in waiting area provide acoustic privacy."

![Diagram](image)

**Figure 9.** Illustration and diagram: "courthouse oriented facing parking lot."

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.

4.1. RETRIEVAL FROM SKETCHING

The premises for this 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 avoid 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 an intrinsic value is 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 links 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 through the diagram catalog.

5. Discussion and Further Work

We believe our simple prototype 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 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 are 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 larger vocabulary of glyphs and spatial relationships. 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 Archie, watching in the background, 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 bubble 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 Domeshke have served as co-principal investigators along with Craig Zimring on the Archie Project, and Domeshke 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 D48-566, and Defense Advanced Research Projects Agency, monitored by ONR under 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


End-User Indexing of Design Lessons

Eric A. Domeshek and Janet L. Kolodner
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
{domeshek | jlk} @ cc.gatech.edu

Abstract: Most complex sorts of design are naturally carried out with reliance on multiple media, and with reference to records and lessons of past designs. Accordingly, we have been building systems that aid designers by making multimedia presentations about existing artifacts easily available. A Case-Based Design Aid (CBDA) contains both basic documentation about prior artifacts, and evaluations of those artifacts with respect to specific issues and tradeoffs. The evaluation-based design lessons are indexed so they can be retrieved by designers concerned with similar issues.

For a CBDA to be most effective, it is best that it be constructable and maintainable by domain experts. In this paper, we look at the implications for a system whose multimedia materials must be indexed by users. We have built several CBDAs and have generalized those systems to produce a CBDA shell called Design-MUSE. This shell has been used in a graduate class on Case-Based Reasoning (CBR) during which student groups constructed several new CBDAs. Here we review what we learned from this initial experiment in end-user indexing.

1. Which Indexes? Who Indexes?

The AAAI ‘94 Workshop on Indexing and Reuse in Multimedia Systems is based on two observations: 1) there are many forms of multimedia data that we would like to be able to retrieve when relevant, but 2) in the current state of the art, we cannot yet automatically analyze these materials sufficiently well to assign good descriptive labels that enable appropriate retrieval. Part of the difficulty, of course, lies in lower level signal processing and interpretation. For us, however, the interesting parts of the problem have more to do with understanding what these media objects are about, when they might be worth retrieving, and how you would formally describe their relevance conditions.

Our operative assumption here is that humans must be involved in creating indexes, but not just for lack of automatic indexing systems. Humans are required in the indexing loop because our systems cannot, of their own accord, invent the right index concepts. Human indexers’ real job is to invent index formats and vocabularies. Our job is to provide those indexers with tools, techniques, and guidance to make their index definition and creation job easier. This paper reports some of our experiences and achievements working towards that end.

Assuming people will be involved in the indexing process, we still have to figure out which people. One important aspect of our work is that we aim to make it easier for domain experts and end-users to create and maintain indexing systems for particular applications. This contrasts with other efforts that assume indexing is a distinct job to be carried on as the full-time effort of a specially trained cadre of “indexers”. An important question is how aiming for end-user indexing affects the kinds of support we must offer for the indexing process.

Our chosen application area is to help designers and those affected by designs, to index and retrieve multimedia presentations conveying lessons about the conceptual design of complex artifacts. In section 2 we discuss the need for such systems. Section 3 presents example systems, emphasizing the lesson materials they contain and their approach to indexing. Section 4 sketches how we support users as they gather and index materials. Section 5 reviews results of initial experiments in which domain experts created systems. Section 6 discusses several changes we intend to pursue based on our experiences.

2. Multimedia Retrieval in Support of Complex Conceptual Design

Application of computers to design is both a profitable area for commercial development and a frequent domain of study in AI. But most efforts to use computers as design tools, or to equip computers to carry out design (or to verify, modify, or debug designs) have focused on detailed design. Thus, for instance, in architecture, most CAD tools are really drafting assistants, but the task of drafting only arises once most of the design has been thought through. We are interested in building tools to help designers with conceptual design -- that early phase of design during which designers analyze the problems they face and formulate their initial strategies.

We want to aid designers with conceptual design of complex artifacts, such as buildings and aircraft. Our approach is based on observation of working designers and insights from the AI paradigm of Case-Based Reasoning (CBR) (Kolodner, 1993). During conceptual design, designers frequently go out of their way to study previous exemplars from the class of thing they are about to design.
Architects may spend time in libraries browsing books and magazines, they may pull files on similar projects from their firm's records, they may actually travel to sites of similar buildings. They seem to be searching for an understanding of the core issues and tradeoffs inherent in the design of a particular kind of artifact, and for suggestions about how to deal with those issues. CBR research suggests that designers are right to take this approach because conceptual design of complex artifacts is so open-ended and poorly formalized that study of previous cases is probably the best source of knowledge available.

One problem with studying precedents to inform new designs, is that it can be hard to get access to useful information. For example, there are millions of buildings out there, and not all of them are interesting, nor is everything about any particular building important. Library sources offer some assistance in finding interesting buildings and in providing reviews of outstanding aspects of those buildings. But library research is still time consuming and the materials are quite spotty, particularly if you are looking for evaluations of particular design decisions that you might want to transfer to a new project. Site visits are even more expensive, and not necessarily much more likely to provide the kind of evaluative feedback that could guide a designer in a new project. Nonetheless, site visits can be important, because they allow designers to walk through a building to see, touch, and hear how it works. In contrast, published reports are usually limited to text and a few photographs or design graphics revealing a bit of how a building looks and how it is put together.

In the Georgia Tech AI group, we are building Case-Based Design Aids (CBDAs) -- tools that provide easy access to multimedia documentation of existing artifacts, and in particular to evaluations of the outstanding good and bad points of those artifacts that can serve as design lessons. By putting such materials on-line, we can not only make access quick and easy, but we can arrange that materials be accessible at more appropriate times. That is, by appropriate use of technology, not only can we provide more useful multimedia evaluative materials, we can also arrange for better indexing than is normally available in current physical libraries.

A typical scenario of CBDA use might start with an architect partially describing a building they are about to design -- say a 40,000 sq-ft public branch library for an urban setting. In response, the system would offer them a list of similar existing libraries, each with basic statistical information, an overview summarizing interesting points about the building's history, and a range of design graphics (such as floor plans, sections, and elevations). These documents are sprinkled with "hot-buttons" that retrieve lesson-bearing "stories", so while browsing the retrieved buildings, the architect's attention can be drawn to important design issues that the buildings illustrate. For example, the floor plan for a particular branch library in Atlanta might have a black dot placed in its children's area tied to a story:

> In Buckhead Library, the children's section is located adjacent to the main circulation desk separated by a curved, free standing wall. Most of the visitors we interviewed stated that they were uncomfortable because the children's department was too far away from the adult lending department to allow them to control their children's activities. Even though the children's section is separate from the adult reading areas, sound dissipates through the room with an echo. Factors contributing to this echo effect are the shape of the roof and its open plan. Some visitors also complained about the acoustical problems occurring at the extreme end of the room.

This story can serve to raise the issue of supervising children or managing noise from children's areas. Stories can suggest strategies for coping with such problems, or, as in this case, they can serve as counterexamples showing how not to implement such strategies. Alternately, this story could serve to critique a proposal already developed by the designer. In a CBDA, having retrieved this story, the architect is also offered both general discussions of the issues it raises, and additional stories of related situations in other buildings. Browsing this issue-based network, the architect can clip any interesting information into an online notebook and add annotations, accumulating a set of exemplars and ideas for use in a new design. With a set of issues in mind, the user could also pose a query to the system, perhaps asking for a wider range of stories about the effect of room shape on acoustics.

3. CBDA Lessons and Indexing

To date we have built two CBDAs: Archie-II for architecture (Domeshek and Kolodner, 1992), and MIDAS for aircraft subsystems (Domeshek, Herndon, Bennett, and Kolodner, 1994). We have also generalized the commonalities between these systems to create a CBDA shell called Design-MUSE (Domeshek, Kolodner, and Zimring, 1994). It was, in large part, the urging of our MIDAS collaborators from Lockheed Aeronautical Systems Company that drove us to construct Design-MUSE so that domain experts (designers) could take much of the responsibility for building and maintaining a CBDA. For Lockheed, it made no sense to contemplate building a corporate memory that required full-time support from AI experts. We expect the economics of many potential applications to work this way, and so we have become interested in the problem of easing system creation and maintenance.

In our CBDA work, the emphasis has so far been on gathering and indexing lesson materials. We have spent less time on the basic design documentation. In addition to the stories about design issues in specific artifacts (like the library story above), lessons include two sorts of general discussions: problems, and responses. A problem is a general statement of issues and tradeoffs that frequently arise in designing some class of artifact, such as a public building or a library. A response is a general statement of some strategy for coping with or resolving a problem. There are usually several alternate responses to any problem. For each response, there may be several stories, some demonstrating success, some failure, and some mixed
consideration in Figure 1 is a common one that arises out of the outcomes. Together, a set of stories provide concrete interpretations of otherwise abstract problem and response statements. The problems and responses, in turn, provide a framework for interpreting the facts of a particular story.

Figure 1 shows part of a screen with a representative set of lesson presentations from Archie-II. Note that lessons are often best conveyed by a combination of text and graphics (or potentially sound or video). A CBDA always offers a cluster of problems, responses, and stories; when space allows, it also displays documentation on the design discussed in the current story. The problem under consideration in Figure 1 is a common one that arises when adults bring children with them to a public building (such as a library). The response suggests that children should have their own supervised space in a public building. The story, which we saw earlier, is a failure of the suggested response, in part because the children’s space is not separate enough, and in part because, given a lack of supervision, it is too separate.

Indexing in a CBDA includes both “relative” and “absolute” indexes. Relative indexes are hard links between presentations, as when a dot on a design graphic serves as a hot button calling up a story. Relative indexes also link problems, responses, and stories. In Figure 1, for example, hard links determine which stories are relevant given that the user has just chosen to view the second response to the displayed problem. The network linking problems, responses, and stories begins to answer the question of how lesson materials should be made accessible to users.

But we still must provide some sort of entry point into this network. In our scenario above, the initial entry was through a search on artifact features. Such descriptions of artifacts are one form of absolute index. We have actually put more time into making it possible to search on lesson interests. For example, the story above could be retrieved based on a query about how the shape or location of a children’s room affects library acoustics. In a CBDA, both problems and stories are assigned absolute indexes -- descriptive labels that can be used as a basis for retrieval when users express their own design interests. Figure 2 shows part of a screen displaying an index for the story in Figure 1.

The index pictured in Figure 2 is representative of our current thinking about what design lesson indexes should look like in general. A similarly structured index framework was used for the MIDAS system. These indexes include two major types of information: 1) the artifact or artifact parts about which we are looking for information, and 2) the issues we are trying to resolve with respect to the artifact. For any complex design task, it is necessary to provide ways to break the problem down into manageable decisions -- parts and issues are two natural ways designers carve their world.

The most obvious and prototypical parts are physical pieces of an artifact. For example, the “children’s room” is part of a “library.” But it also often useful to pick out parts that may be physically distributed, yet identifiable as functional subsystems within an artifact. In a building, the “acoustical” system is a good example; many separate physical pieces spread throughout a building can all be chosen and coordinated for the effect they have on noise and sound dispersion.

Issues may reflect decisions about the structure of the artifact or consequences for the functioning of the artifact. When considering consequences, it is necessary to keep in mind that an artifact typically has many concerned constituencies, each with their own perspective and goals: a library, for instance has to satisfy not only its users, but also the staff (including librarians and maintenance people), the relevant government officials, and the community. The wide range of stakeholders becomes even more apparent once you consider an artifact over its entire life cycle: aspects of a library design can affect the contractor who constructs it, those who renovate it, and of course, the architects themselves.

4. Tools for Building CBDAs

Design-MUSE is intended to make it easy both to define an indexing system and to assign indexes to individual lesson presentations. Defining indexes and assigning indexes are treated as two distinct levels of privilege in the system; only some of the users empowered to enter and index materials might be allowed to redefine the indexing system. An indexing system is defined as a set of frame types whose fields are each restricted to take
fillers of a particular type. The allowed fillers may be other frames, or they may be primitive elements of some class (much like the elements of Pascal's enumerated types). An index, then, is an instance of a frame filled out with enough detail to label a problem or story, or to express a user's interests for retrieval.

The system provides a complete interface for defining classes of primitives in terms of a micro-feature encoding (Hinton, 1981). This encoding affords a basis for nearest neighbor matching during retrieval. Primitives may be defined, redefined, and renamed dynamically. Attempts to delete primitives already in use are flagged. Synonyms and overlapping meanings are easily established. Primitives are automatically organized into sets of hierarchical menus based on meaning overlap. In contrast to the situation for primitives, the interface for defining frame types in terms of slots with type restrictions is not yet fully implemented, so some resort must currently be made to editing underlying data files.

To ease construction of effective indexing systems for lesson retrieval, we would like Design-MUSE to provide new users a suggested starting point for indexing. We aim to discover a reasonably general "Design Index Frame" (DIF) that we can build into the system. This idea of a DIF is related to the earlier proposal of a "Universal Index Frame" (UIF) (Schank, et al., 1990), however, the intent here is to restrict applicability to indexing lessons on conceptual design of complex artifacts; we also assume that there will be variants of the basic DIF. Experiences reported in the next section give us some initial feedback on how much commonality we can expect to achieve. As we will see, while we have enabled successful construction of CBDAs, our initial DIF proposals have done better at characterizing the content of indexes than at supplying a useful structure for that information.

In addition to supporting index system definition, Design-MUSE also provides a complete interface for assigning indexes to presentations and for creating search cues. Users may create, delete, or copy indexes, and they may add and delete fillers, all using a simple menu-driven interface as shown in Figure 2 above. The menus of primitive filler choices are those generated by the system from the definitions of sets of primitives. We believe it is important to the usability of the system that the same interface works both for posing queries and for indexing stories or problems.

Shared interface elements for browsing and for entering data extend beyond indexing to many other aspects of the system. For example, new texts are entered by typing (or cutting and pasting) directly into the same windows and panes that are used for display. Links among presentations are created using the same widgets that later traverse those links. The goal throughout is to make information access easy, information display clear, and information entry simple for anyone who knows how to use the system for browsing. This principle cannot yet be applied to non-text media forms, as the program does not include an interface for creating design graphics (but see the discussion of work in progress in Section 6).

5. Experience with End-User Indexing

This section discusses issues uncovered as users with little AI training attempted to build CBDAs in their own areas of expertise. While MIDAS was built primarily by Lockheed personnel, it was built at the same time as Design-MUSE and its indexing system was put together with significant assistance from our AI group. Six other CBDA-like systems have been, or are being constructed by people whose primary expertise is in domain content rather than indexing techniques. The first five were group term projects in a graduate CBR class; they were all constructed using Design-MUSE. The sixth is a masters thesis project, still in progress; it is being built in an alternate tool set, primarily to avoid programming language conflicts. The systems, with their domains and primary tasks are as follows:

- **Next-Archie**: A revised-extended version of the Archie CBDA for building design, aimed especially at student architects. (Design education)
- **Composites**: An advisor to help with decisions about replacing metal parts of aircraft with parts made from composite materials. (Re-design)
- **DSP-Aid**: A guide to formulating design problems in the "Decision Support Problem" (DSP) methodology so they are suitable for analysis using DSP software tools. (Design process)
- **GT-SCARES**: An on-line trouble log to help in diagnosing and coping with satellite failures. (Diagnosis and debugging)
- **Dream-Car**: An advisor for consumers considering purchase of a new car. (Situation assessment)
- **CMS-Browser**: A repository for analyses of several independent satellite command management systems in a common framework. (Design re-use)

These systems are not all CBDAs as we had originally conceived them. While all aim to support decision-making with respect to complex artifacts, they vary quite a bit in terms of what decisions they support. Some have little to do with design, per-se.

To create their new systems, the students were told to propose use scenarios and to think carefully about the decisions their users would have to make and how they would make them. Each group had at least one student with experience in the domain. They were instructed to collect stories that bore on those kinds of decisions and to analyze under what conditions past decisions might be repeated. After analyzing a half dozen cases (which might yield a larger number of decision relevant stories), they were to come up with some initial indexing proposals.

When we gave each student group their copy of Design-MUSE, we provided an initial index framework with five dimensions: part, system, issue, stakeholder, and life cycle. We also supplied a basic set of primitives for the fillers of the stakeholder and life cycle dimensions. None of the groups stuck with this initial framework.
This was not terribly surprising in the case of systems such as GT-SCARES or Dream-Car which were not primarily concerned with aiding designers. It is more interesting in the cases of systems like Composites and DSP-Aid. (Next-Archie and CMS-Browser stayed much closer to the initial framework).

In the design of an indexing system for a CBDA, there are pressures from at least two different directions: 1) Sufficiency: the system must allow the expression of features that indicate relevance of applicable advice, with enough discrimination to rule out less relevant information; and 2) Simplicity: the system must be obvious and easy enough for an untutored domain expert to express what they believe matters about a situation. Simplicity militates for a small number of index dimensions; sufficiency may suggest including more. Simplicity pushes towards the inclusion of only a relatively small number of general and coarse distinctions; sufficiency may require many more fine domain and situation specific distinctions. Another question in index system design is the extent to which the various dimensions and their fillers ought to be orthogonal -- whether redundancy should be rooted out, so there is always only one way to express any description. Here it is not clear what constitutes the simpler system, as a clearly orthogonal framework may be easier to understand, but may be more cumbersome to use.

All of the systems tended to pay more attention to simplicity than sufficiency, in part because the small corpora (usually about 50 items) in these systems did not require hugely elaborate indexing to achieve sufficient discrimination. The DSP-Aid and Composites systems both used relatively many dimensions (seven each), but each dimension was sparsely populated with only a few coarse descriptors (in DSP-AID the average was only three choices per dimension). On the other hand, the Next-Archie and Dream-Car systems had fewer dimensions (five each), but not only were those dimensions populated with broader range of vocabulary items, they were also developed to a much deeper detailed level (often with dozens of choices). Dream-Car and Next-Archie did a better job of establishing orthogonal dimensions that clearly described different aspects of a situation; that was less clear in the case of Composites.

Some examples should give a flavor of the kinds of idiosyncratic dimensions and fillers that the student groups came up with (ellipses indicate when the lists of fillers was longer than the few listed presented here):

DSP-Aid:

| Type         | e.g.: Structural, Management, Composite, Mechanical ...
|--------------|------------------------------------------------------
| Objectives   | e.g.: Single, Multi-Level, Multi-Objective          |
| Information  | e.g.: Imprecise, Statistical                         |
| Decision-Var | e.g.: Discrete, Continuous                          |
| Subsystem-Rel| e.g.: Coupled, Independent                          |
| Analysis-Tools| e.g.: Yes, No                                      |
| DSP-Issues   | e.g.: Compromise, Fuzzy, Bayesian                   |

Dream-Car:

| Car-Make     | e.g.: Japanese, Toyota, Tercel ...
|--------------|------------------------------------------------------
| Issues       | e.g.: Safety, Reliability, Fun, Economy, Comfort ...
| Features     | e.g.: Security-System, Audio-System, Safety-Features ...
| Price-range  | e.g.: Cheap, Mid-Range, Expensive ...                |
| Buyer-profile| e.g.: Family, Hauling, City-Driving, Income ...      |

When we go back to analyze the dimensions and fillers in the students' indexing systems, as well as their use of those systems, we find patterns that do not differ as much from our original proposal as at first seemed the case. There are two different types of CBDA indexes: artifact indexes and lesson indexes. Artifact indexes are composed of situation descriptors and artifact features. Lesson indexes are composed of artifact parts (including physical and functional parts) and design issues (differentiated by stakeholder and life-cycle). Most of the students' dimensions fit into one of these categories. However, because of shortcomings in the implementation of Design-MUSE, students crammed everything into the lesson indexes.

In the DSP-Aid for instance, the students primarily reserved the type dimension for indexing "background stories" that were about an entire design rather than about any particular decision; were the shell in better shape, they might have treated this information as design documentation and treated type as a dimension of their artifact indexes. Many of the DSP-Aid dimensions would have made more sense as descriptors of the problem context or a proposed partial solution (that is, as artifact index descriptors), rather than as descriptors of lessons directly.

In the Dream-Car system, car-make and price-range match artifact index dimensions included in Archie-II (building-type, and cost). However, the more novel and complex dimensions, issues, features, and buyer-profile all map onto our proposed lesson index dimensions (issue, part, and stakeholder).

Beyond prodding us to complete the Design-MUSE implementation, study of the students' systems reminds us that artifact indexes can usefully serve as a context for interpreting lesson indexes. That is, it may be appropriate when searching for lessons, to consider only those bearing on artifacts known to fit the user's artifact index.

Though we can see how the students' indexing systems broadly fit an expected pattern, they were the result of independent analysis, and the quality and depth of that analysis varied considerably. The more comprehensive systems were those whose builders did not settle for the minimal set of dimensions and fillers that covered their data. They got more interesting results by going back over their initial index frameworks looking for generalizations. They often regrouped descriptors into new dimensions and filled out the resulting categories with related descriptors that fit the revealed pattern even when they did not have
stories in hand that needed those vocabulary items. One effect of building a system like this is to discover something about the structure of your domain. One effect of using a system like this is to learn something about how experts have found it useful to carve up and describe the domain.

It is interesting to quote what one set of students wrote about their experience generating the indexing system for the Dream-Car CBDA:

"In order to understand our system it will be helpful to consider its development chronologically. From the start of the project we had created a number of cases. After digesting these cases, we began to see a core vocabulary emerge. We then took a bottom up approach, which means that we looked for terms that would describe several elements at once. So we came up with words like: price, salary, money, Toyota, Japanese, comfortable. We then tried to ascertain logical ways in which these terms might be grouped. After a while the fog began to lift and we were able to identify abstract dimensions such as "Buyer Profile" for describing personal conditions and situations in which a buyer might find herself. At this point, a top-down approach was taken and we were able to see specializations of these abstractions that we had not seen from the start. Vocabulary items such as "personal constraints" emerged from this top down approach. We learned from this experience that in order to index the stories, we had to first have a basic concrete knowledge of our domain. When we had climbed high enough to see with some perspective what lay below, we were able to see what was missing. In the end, we were left with a rather complete descriptor of the car domain."

We found it interesting how adaptable the Design-MUSE shell proved to be. Not only was the design process group able to fit the pattern, but even the non-design groups were able to put together systems that made sense. Of course a significant portion of the credit must go to the ingenuity of the students, who were also able to work around some of the more glaring gaps in the version of Design-MUSE they were given to work with. Still, there seems to be general usefulness in providing easy access to collections of multimedia presentations about specific examples of problem solving situations, and linking those to more general discussions of principles that bridge and organize the examples. Further, Design-MUSE seems to provide a flexible framework for quickly constructing such systems.

6. Discussion: The Changing Face of CBDAs

CBDAs were originally conceived as a way to demonstrate useful application of CBR principles, and as a way to drive further CBR research on building, indexing, and retrieving from large case libraries. While we believe we are still on track for both of those goals, it appears that they may not be as tightly coupled as we at first assumed. In particular, some designers have suggested that useful CBDAs could be built with relatively small sets of lessons -- systems with on the order of hundreds of stories might be fielded to good effect. This is a far different picture than one in which an endless stream of varied materials flows into a library and the entire information flood must be indexed for a wide variety of possible uses. Still, we expect learning to manage smaller systems will help us with larger ones to come.

Even assuming systems built with relatively small numbers of stories, one issue that requires more attention is how to support domain experts in generating and linking useful generalizations (problems and responses) to collected stories. We need to develop better methodologies and tools for this critical analysis task. To some extent, the common indexing framework for stories and problems starts to address these issues. Our experience to date, however, is that much material may accumulate before a settled indexing system evolves, and we do not want to wait until late in the game to form generalizations and connect them to examples. Perhaps a better start on indexing will help address this problem.

We have one extension to the initial index frame already in mind: we propose to include explicit outcome relevance information. For instance, it is important to know whether a story illustrates a success or failure (and with respect to which design intentions); it is important to know whether a story talks about a way to avoid a problem, a way to fix a problem, or a way to compensate for a persisting problem. Including this information in our absolute indexes should make it easier to build useful relative indexes for the lesson network. It should also support focused search for particular kinds of lessons.

In addition to looking for generality and stability in the conceptual indexing of design lessons, we have also started looking for recurring patterns in our graphic media that can be used for indexing. Gross, Zimring, and Do, (1994) report on some new work, related to Archie-II, in which architects' sketches were analyzed for commonalities and in which structure extracted from free-hand diagrams was used to retrieve information on similar designs. This line of research requires our CBDAs be linked to graphic input tools that begin to provide a design environment. Again, one of our goals will be to ensure that the interface for graphic creation extends gracefully to an interface for graphic indexing.

Ultimately, the success of Design-MUSE will depend on our ability to embed useful conventions in the system so that new users intuitively understand and apply what we have learned about CBDAs. At a basic level of system usability, we believe we are reaching the point where we have a robust and comprehensible system. Though some features remain to be implemented, and others will be reimplemented based on our experience, users have been able to drive the system successfully. The next step, however, is to do a better job at communicating to users what kinds of materials and indexes ought to be in

---

1From the final report on the Dream-Car system by Tal Cohen and Todd Griffith.
the system. While we have comprehensive documentation on the mechanics of the system (soon to appear also in an on-line help system), style sheets on how to gather and prepare data are still under development. Guidance on how to index the resulting presentations is even further in the future. As our initial experiments show, task and domain variety beget index variety. Only through experience will we arrive at a better characterization of a space of tasks and domains that we can easily support in system like Design-MUSE, right off the shelf.

Acknowledgments

This paper would not have been possible without the effort and perseverance of the students in Prof. Kolodner’s CS-7332. In her work on satellite command management systems, Jennifer Ockerman was especially helpful. In addition, we must also thank the many contributors to Archie and MIDAS: Craig Zimring, Ashok Goel, Richard Billington, Anna Zacherl, Vijaya Narayanan, Marin Simina, Ellen Do, Hassum Khalil, Ali Malkawi, Ameen Farooq, Osman Ataman, Marcia Herndon, and Andrew Bennett. This work has been supported in part by the Defense Advanced Research Projects Agency, monitored by ONR under contract N00014-91-J-4092, and by the Lockheed Aeronautical Systems Company. All views expressed are those of the authors.

References


Scaling Up is Hard to Do: 

Experiences in Preparing A Case-Based Design Aid Prototype for Field Trial

Eric A. Domeshek1
Craig M. Zimring2
Janet L. Kolodner3

Abstract

We seek to improve the conceptual design process for complex artifacts such as buildings by offering designers access, through an online hypermedia library, to documentation and evaluations of existing projects. We have constructed Archie-II, a prototype of such a Case-Based Design Aid (CBDA) for use by architects. After several iterations, we are now nearing our first field trials of Archie-II; students in the Georgia Institute of Technology's College of Architecture will use the system in studio classes. This paper discusses our experiences preparing for these trials, emphasizing lessons learned about building CBDAs.

Case-Based Design Aids

Design of complex artifacts is a complex business. Consider, for instance, building design: an immense number of decisions are required to fully specify any moderate to large sized building, and managing the information required for such design processes is a daunting task. Not surprisingly, computers are being applied to information management

1Research Scientist, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280.
2Associate Professor, College of Architecture, Georgia Institute of Technology, Atlanta, GA 30332-0155.
3Professor, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280.
problems in design with ever greater frequency. Modern CAD tools record not only the specification of form and structure with high precision (often high enough to support meaningful simulation), but also store details of parts, materials, costs, and simulation results; such data feeds on to estimating, planning, and scheduling systems that capture and generate still more information about the evolving design.

Most of these tools are aimed at managing the information flood during the detailed stages of design. Computer-based tools to aid in conceptual design remain scarce. This earliest phase of design, however, has its own severe information requirements, and given the importance of early decisions to the course of a project, investments in computer aids for conceptual design could pay high dividends. We have been developing a class of computer-based tools to aid in conceptual design of complex artifacts. The aid we aim to provide addresses one of the critical information needs during conceptual design: easy access to lessons from past experience. To recognize potential issues, to sort out those that are relevant, and to select strategies for addressing key issues, designers need information about existing designs, and most especially, evaluations of what worked well and what went badly in previous cases.

Our recognition of this need, and our approach to satisfying it derive from research in two disparate fields: the artificial intelligence paradigm of case-based reasoning (CBR) (Kolodner, 1993), and the architectural practice of post-occupancy evaluation (POE) (Zimring, 1987). The basic idea in CBR is that much of expertise is based on the ability to recall and apply past experiences. The basic idea of POE is that building design can be improved by paying attention to the needs of users and other stakeholders as revealed by empirical evaluations of existing buildings. Our case-based design aids (CBDAs) are on-line hypermedia libraries of design data and evaluations. The goal is to make this information easily available to designers when it will do the most good. The systems lets users pose queries to retrieve prior designs and lessons learned from those designs as they work their way through new design problems. An experience drawn from the library can raise design issues, point out pitfalls and opportunities, or suggest possible solutions.

CBDAs organize stories about existing designs in a network linked with more general statements of problems and problem responses. Figure 1 shows part of the “Lessons” window from the Archie-II CBDA for building design. The three panes display a problem (a general statement about some design situation that often leads to difficulties) a response (a general strategy that may help resolve the problem) and a story (a discussion of some particular building and how it coped with the problem). Alternate responses to problems, successful and unsuccessful attempts to apply responses are all easily available. Each story displayed
brings with it graphics that focus on the appropriate parts of whichever building is under discussion; these graphics displayed in the "Designs" window (not shown here) provide context for story interpretation.

Project History

The Archie project is an ongoing collaboration among members of the College of Computing and the College of Architecture at the Georgia Institute of Technology. In discussions dating back to 1990, the group formed around shared interests in design, cognition, and improving conceptual design practice. The original Archie system (Pearce, Goel, Kolodner, Zimring, Sentosa, and Billington, 1991) introduced the idea of supporting architects by providing easy access to information about existing buildings. Since then, Archie-II (Domeshek and Kolodner, 1992, 1993) has been under continuous development as we refined that idea and worked to make it a reality. Other similar systems have also been developed, such as AskJef (Barber, Bhatta, Goel, Jacobson, Pearce, Penberthy, Shankar, Simpson, and Stroulia, 1992).

The path from concept, through story-board, demo, prototype, and finally the current robust prototype has been long and challenging, spanning four years and four different development environments. The current incarnation of Archie-II is an evolving library of building data and analysis managed by a general CBDA shell. The shell, called Design-MUSE, provides data entry and browsing facilities. It was built as part of our first attempts to generalize this work beyond building design. With Lockheed Aeronautical Systems Company, we developed the
MIDAS CBDA for conceptual design of aircraft subsystems (Domeshek, Herndon, Bennett, and Kolodner, 1994), and with it, the tool that became Design-MUSE. Design-MUSE and our CBDAs all run on the Macintosh.

We are currently targeting Archie-II for use in studio classes at the College of Architecture. Our pedagogical aims include both presenting students with an interesting range of past designs, and reinforcing the habit of critically assessing designs by consistently linking the results of design with underlying design intentions. For instance, the problem statement in Figure 1 identifies a pair of design intentions in conflict (easy access to restrooms vs security for books) and the story assesses how well a particular design resolved this conflict. The system would also present graphics illustrating the layout of this library and let the user explore related issues that might touch on related features of the design.

Towards a Fielded Prototype

In the last nine months, since June 1993, Archie-II has matured considerably with regard to content, organization, presentation, indexing, and interface. We will discuss each of these aspects in turn.

Content

At the end of the last academic year, Archie-II's knowledge was confined to documentation and evaluations of two courthouses, plus a collection of courthouse design guidelines. Documentation was limited to floor-plan graphics. The evaluations comprised about one hundred stories, many focusing on detailed failings of one of the two courthouses. We have made two major advances: first we have increased the number and variety of buildings represented in the system; second, we have tightened our notion of what constitutes an interesting evaluation story for conceptual design. Many detailed stories that only addressed failings with respect to single issues have been dropped in favor of stories that deal with more strategic tradeoffs among multiple goals.

Gathering this additional data has not been easy, nor are we yet totally satisfied with the quality and quantity we have accumulated. A group of five graduate students, mostly trained architects, conducted a series of building site visits from July through December of 1993. Many buildings had to be visited repeatedly in attempts to gather floor-plans, distribute and collect questionnaires, interview users, observe use, take photographs, and so on. As already suggested, we aimed, in this field work, to gather stories about strategic tradeoffs that would be of use and interest during conceptual design. Unfortunately, our methodology, which included providing students with suggestive problems observed
in other buildings, proved too directive. Much of the data gathered bore superficially on the sample problems, while other interesting issues (plus the details to support discussion of those issues) were missed.

As of one month before trials are scheduled to commence, Archie-II contains information drawn primarily from 10 different buildings -- four courthouses and six libraries. All but one have been visited and evaluated by members of our project team; the exception is our first courthouse described in a published report (Building Diagnostics, 1988). The lesson information comprises about 70 problems, 100 response, and 200 stories, supported by sketches, diagrams, and photographs. We are assembling basic design documentation for these buildings, including floor-plans and summary statistics, but this data is not yet entered.

Organization

Until September, Archie-II organized its lesson information into two major classes of presentations: stories and guidelines. Stories, as in the current system, gave detailed evaluations of specific buildings. Guidelines gave general information about design issues; essentially guidelines combined discussion of problems and responses. We decided to split guidelines into separate problem and response discussions based on feedback from architects we brought in to evaluate the developing system. They felt that guidelines often read too much like authoritative statements of what "ought" to be done, when in fact they were intended only to be suggestive and to indicate a range of possibilities.

Splitting guidelines into problems and responses built into the system's interface the assumption that there is nearly always more than one way to resolve a problem. The split helped drive us to refine our problem statements so that they identify the active design goals. This in turn helps reinforce connections between analysis of goals and synthesis of solutions. This new organization also makes it easier to add new problems and responses as we discover them in studying new buildings.

Presentation

We have, on and off for two years, attempted to develop standards for presentations in the Archie-II system. In the last three months, we have generated a set of style-sheets detailing the content, organization, and style for problems, responses, stories, lesson illustrations, building data, graphics, and summaries. After several cycles of presentation preparation, style-sheet proposals, presentation revision, and style-sheet refinement, we believe we are converging on workable standards. As with the initial data gathering, however, the fact that this is still our first time through the process means that we often discover that we have not
got the material we really want once we figure out what that would be. Our materials were gathered over a year or two, using varying methods and standards, by a shifting group of students (none of whom are native English speakers). We are now in the midst of a final editing cycle so we can achieve uniformity and high quality in the final presentations.

Effective use of graphics, both to illustrate lessons and to set them in context, is one of the major requirements for any CBDA, but especially for a system aimed at architects. When it comes to illustrating lessons there are two major issues we have identified so far. First, the kinds of illustrations that make sense for general texts, (problems and responses) is different from what is appropriate to a specific texts (stories). General illustrations can often be schematic sketches including only the minimal lines and labels that make the point. Specific illustrations will often be detailed diagrams or even photographs that show exactly how things were done in some building. The second point bears most strongly on story illustrations because they may benefit from greater detail: what kinds of details matter? Consider, for instance, the use of photographs: a photo will almost never be appropriate for illustrating a circulation issue because you usually cannot see enough of the building from one vantage point; on the other hand, a photo may be the best way to illustrate localized details; and when it comes to illustrating the “feeling” of some space, effective architectural photography is an art form all its own.

Indexing

Design-MUSE provides two search facilities: artifact (building) retrieval based on user specified features, and lesson (story or problem) retrieval based on user specified design interests. Artifact search was implemented back in September, but hasn’t been tested extensively because we are still gathering the basic data about our buildings. Lesson search has been part of the system from the very beginning, and the index language used to express design interests has been under constant development and refinement throughout the Archie-II project. Only recently, however, have we had a significant corpus of stories in the system, so we are about to enter a phase of intense experimentation with index content. We have, for instance, recently developed a more detailed analysis of design goals, so one of the tasks before us right now is to fold this theory into the implemented indexing system.

Interface

The capabilities of Design-MUSE have been improving over the last half year. Separating problems and responses, and implementing artifact search are two examples mentioned already. Many other aspects of the shell and its interface have been modified and refined based on
feedback from team members, observers, and actual users. Members of the MIDAS team have been quite helpful: Students in a graduate course at the College of Computing this quarter are also providing useful feedback as they attempt to use the shell to construct their own CBDAs. A more formal usability study is currently in the works. More extensive use and evaluation will follow during the Spring quarter trials.

Archie-II inherits from Design-MUSE a relatively straightforward interface that supports basic browsing capabilities for end users, as well as more advanced data entry and system modification features required by those building the system. Several features required for the trials remain to be implemented: the ability for students to maintain their own files of clippings and personal notes, related facilities for saving and printing such files, and an on-line help system. We expect these end-user features, plus updated documentation to be ready by Spring quarter.

Lessons Learned

We have faced significant challenges at all stages of this project: gathering documentation on a large set of buildings, surveying those buildings for outstanding positive and negative aspects, relating those aspects to design intentions, preparing presentations of this information, organizing those presentations for effective retrieval or browsing, and constructing a robust friendly interface to the material. In all cases, we must document our mistakes and codify those techniques that ended up working. Design-MUSE itself embeds many of our solutions touching on the structure and interface of a CBDA, making them easy to replicate in new domains. But our experience to date also points out the absolute necessity for tighter data gathering methodology and settled style sheets to serve as guides to what will be useful data.

Two points on data collection are worth making here: first, site visits must be structured, but note-taking must be broad enough to support late identification of problems, responses, and stories; second, stories are most useful when they let informants speak with their own voices and evaluate a design from their own viewpoints. How well we absorb these and other lessons will determine our success in bringing diverse user viewpoints to our design students.

Acknowledgments

The Archie project has involved many people, and benefited from the contributions of many others not directly involved. In addition to Ashok Goel and Richard Billington who helped shape the project at the
earliest stages, we have been fortunate to work with CS students Anna Zacherl, Vijaya Narayanam and Marin Siminis, and architecture students Ellen Do, Hassum Khalil, Ali Malkawi, Ameen Faruck, and Osman Otaman. This work has been supported in part by the Defense Advanced Research Projects Agency, monitored by ONR under contract N00014-91-J-4092. Design-MUSE was supported in part by Lockheed Aeronautical Systems Company. All views expressed are those of the authors.

References


Supporting Case-Study Use in Design Education:  
A Computational Case-Based Design Aid for Architecture  
Craig Zimring1, Ellen Do1, Eric Domeshek2, Janet L Kolodner3.

Abstract

Cases play a central role in architectural design education. Many architectural journals, books and magazines are based on case studies of buildings, and instructors commonly use cases in lectures and studio teaching. These cases, which are often descriptions of the form, materials and design strategies of significant buildings, are used by students for a variety of purposes. They help elaborate and refine stated problems, suggest and refine design solutions, help evaluate a completed design, suggest possible alternative solutions. However, appropriate cases are scattered across the print literature and may be difficult to find.  

Perhaps more important, the ways in which most current cases are written and made accessible to students does not exploit their full potential as pedagogical tools. In this paper we describe “Archie,” a “case-based design aid” (CBDA) that provides architecture students rapid and flexible access to evaluated multi-media design cases. A CBDA is a computational aid that provides novice and experienced designers access to cases that they can then adapt to solve their current design problems. Based on the theory and technology of case-based reasoning, a CBDA uses the computer to store and access cases and allows human designers to do their own adaptation.  

Archie is aimed at aiding teaching of conceptual design. It is intended to help students: 1) understand the intentions of the range of stakeholders who have an impact on a...
building project, such as designers, users, clients, and builders; 2) develop more explicit statements of goals and criteria for success of their designs; 3) make links between general goals and specific design solutions; 4) link written statements of goals and needs to graphic representations; 5) keep design problems open long enough to allow students to explore an appropriately wide range of possible solutions.

Archie: Content and Browsing

Archie is a computer tool aimed at supporting teaching of conceptual design and at aiding design. It is written in Common Lisp on Macintosh, and is based on a theory and technology called “case-based reasoning” that suggests that much effective problem-solving is based on retrieving and adapting appropriate specific past experiences (Kolodner, 1991, 1993). Archie provides students access to cases based on post-occupancy evaluations of occupied buildings (Friedmann, Zimring & Zube, 1978). Cases in Archie are structured as descriptions, problems, design responses and stories.

- **Descriptions** are the most common architectural presentation formats in architectural books and magazines. In Archie, these are available as building plans, sections, elevations, sketches and texts.

- **Problems** in Archie are dilemmas to be solved. We believe that it is important for students to understand that most design decisions attempt to resolve conflicting requirements. Problems in Archie are usually represented as two or more relatively general intentions, with an implementation that fails to resolve both intentions. The program then allows students to explore responses that will resolve both intentions.

- **Design responses** are general strategies a designer might consider for reconciling the intentions.

- **Stories** are brief descriptions of how the problem or solution has played out in a specific building.

Graphic material (illustrated/annotated plans, photographs, and drawings, and even quick time animation sequences when applicable) are associated with each text item. Each problem and story is indexed using a set of features (index terms, issues, systems, etc.) developed specially for the Archie case library. Related items are also cross-linked for browsing.

In developing Archie, we observed students and professional designers doing design and discussed pedagogical strategies and problems with design instructors. We have attempted to provide a high degree of flexibility for users of the system to access information to fit their personal styles and specific tasks. However, we also provide “scaffolding” by linking related items. Users can find an appropriate case in two major ways: index-based browsing and hypertext-based browsing.

- Using the **indexing system**, a user can search for a given artifact (building or part of building) or search for a problem, story, or response. These indices reflect characteristics of the design such as size, architect, location, or use-type of the building, as well as the content and pedagogical point of the problem or story, such as stakeholder type or mechanical system. For instance, if a student knows she will soon be doing a studio problem involving a child-oriented public library, she can search for all stories or problems related to children in libraries. In browsing these stories she soon finds some important dilemmas that need to be resolved in her design. For instance, adults who bring their children to the library often like to maintain visual supervision over their offspring, but if the children’s area opens directly onto the adult reading room, noise from the children may disturb other readers. Later in her design process, she may search for information about a specific feature such as circulation desks, or may search for descriptions of a specific library that she was interested in learning about.

- With **hyper-text links**, the user can move directly from design descriptions to stories, from stories to problems, and between problems and responses. For instance, the user described above, having found the problem of children’s noise disturbing adult readers, can move directly to descriptions of two or more response strategies (such as using separate rooms, acoustic treatments, or glass walls) and how these responses have worked in stories of actual buildings. These stories are linked to design representations such as floorplans.

A designer’s search may also take her in unexpected directions. For example, in Figure 1, below, a student who is interested in learning about problems related to bringing natural light into the interiors of library buildings may encounter stories about courthouses that have grappled with the same problem. The system then allows her to copy the appropriate indices for the specific story to allow further search, as is illustrated in the upper part of the figure. Or, she can use the hypertext links between story and design to explore a specific courthouse case that in fact has many problems shared with libraries, such as maintaining security. This is shown in Figure 2.
Successful design solutions resolve conflicting intentions, and that problems typically have several potential strategies for resolving them. Text is always accompanied by graphics.

---

**Figure 2:** Stories are linked to a representation of the design such as a floorplan or elevation to allow students to put stories in context of the overall design solution.


Case Libraries in Support of Design Education: The DesignMuse Experiences

N. Hari Narayanan and Janet L. Kolodner
EduTech Institute, College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

Abstract

A major part of the reasoning designers do involves accessing old cases and using the lessons learned in those situations to address new problems. DesignMuse is a shell for creating on-line case libraries with built-in browsing capabilities that make cases available to design students in a natural way while they are designing. Archie-2 is the most extensive case library built with DesignMuse, and has been used in Georgia Tech's design studios. Archie-2 provides flexible access to multimedia case representations of buildings: libraries, courthouses and skyscrapers. Susie is a student-built library containing cases about sustainable technology and development. These two case libraries illustrate the two ways in which we believe case libraries can enhance learning in the classroom: (1) by using the case library to search for, analyze, compare and contrast cases that are similar to the problems students are solving, which will improve their problem solving skills, and (2) by constructing cases and building a library after doing research in a domain, which will help them learn domain knowledge. This paper describes our classroom experiences with these two DesignMuse-based case libraries and work in progress on enhancing various functionalities of these libraries based on feedback from students and informal classroom observations.

1. Introduction

Design is an activity that requires not only knowledge of facts and fundamentals, but also the spark of creativity and insight. In order to generate innovative solutions to design problems, expert designers often turn to the history of failed as well as successful designs as an invaluable information resource. For novice designers, such information is even more valuable because for them it can serve not only as a source of ideas but also teaches them about the impact that good and bad designs have had in the past, different perspectives that were brought to bear on trend-setting designs, and the plethora of issues that arise during design. Thus, previous designs can be a rich source of ideas and inspiration in the early stage of design when the design problem is still open-ended and evolving. Another fact to note is that design in the real world has increasingly become a multi-disciplinary group-oriented process in which multiple perspectives converge - aesthetic, ergonomic, economic, technical and social, to name a few. Often, successfully addressing issues that arise during design requires an understanding of the interactions between those issues and an ability to arrive at appropriate tradeoffs. All these strongly suggest that collecting, organizing, indexing and presenting design cases structured in a way that highlights the relevant issues involved and lessons that can be learned from them, and making these available as an on-line library of design experiences, can be a very powerful aid for both design students and expert designers. Therefore, design case libraries and case library authoring tools have become an important part of the design education initiative of EduTech Institute, Georgia Tech. The use of case libraries for education derives from research on case-based reasoning [5], the use of past experiences to solve current problems. The central idea in case-based reasoning is that cases facilitate the solution of new problems by suggesting applicable past solutions, pointing the way out of quandaries, allowing potential failures and errors to be anticipated and avoided, and focusing attention on relevant issues.

We believe that design cases can be used in two ways to help students become better designers and learn domain knowledge well: (1) Analyzing past cases while solving design problems can help

students do design better in a variety of ways: cases illustrate how design problems have been solved in the past, provide warnings about potential pitfalls, focus attention on significant issues, suggest potential solutions, and guide adaptation of earlier solutions to fit the current problem; (2) Constructing new case libraries for others to use helps students acquire deep knowledge of a domain - since cases are actual instances that illustrate the application of knowledge, knowledge thus learned should be transferable to new problems. Archie-2 and Susie are two case libraries which illustrate these two roles. In this paper we describe our classroom experiences with these two libraries and current work on enhancing various functionalities of these libraries based on feedback from students and informal classroom observations.

2. DesignMuse and Case Libraries

DesignMuse [3] is a case library authoring tool that allows easy construction of structured, indexed and searchable databases of analyzed case studies for students to learn from. DesignMuse builds functionalities for indexing, organizing and presenting case information into the case libraries created from it. It also provides graphical user interfaces for specifying search probes, for presenting case information and for modifying/extending the case library. Cases are structured in terms of stories, problems, responses and design overviews. These four pieces of information are presented in four panes of a case presentation window as shown in Figure 1. Stories help students discover which issues they should be considering and help them to anticipate the results of carrying out proposed solutions. Stories are associated with general problems and responses they illustrate. The presentation format - a story flanked on the left by the problem and on the right by the response - is intended to help students take the abstraction step from specific stories to the general problem-solution pairs that the stories illustrate (and vice versa). Design overviews, appearing in the lowermost pane of the case presentation window, provide appropriate context for stories, problems
and responses. Hypertext links between design overviews and stories, and among stories, problems and responses allow flexible navigation and browsing.

DesignMuse has been used both by experts (faculty, graduate students and working engineers) to construct case libraries for use in design studios and by novice students to construct prototype case libraries. Archie-2, described in more detail later, is an example of the former. DesignMuse was also used by students in a graduate level case-based reasoning course in two quarters. In the first quarter (Winter 1994), groups of students produced six different sample case libraries: (1) a building design case library, (2) a re-design advisor that provides advice about replacing metal aircraft parts with composites, (3) a case guide to formulating design problems in the decision support problem framework, (4) an on-line failure catalogue for diagnosing satellite failures, (5) an advisor for car purchase decisions, and (6) a satellite command management system. During the second quarter of use (Winter 1995) DesignMuse was used by three groups of students to create three parts of a single case library on the topic of sustainable technology and development. One group focused on cases of industrial pollution, another on issues of sustainable development and resource management, and the third on industrial accidents. Susie (SUStainable technology Interactive Education) is the case library that resulted from merging these three parts.

Archie-2 [2] is a building design case library. It provides flexible access to multimedia representations of building designs. Each building has been evaluated in a post-occupancy evaluation, and is represented as a series of plans/montages, design issues/problems that came up during its design, construction and use, appropriate responses to those issues/problems, and stories that illustrate how various design issues/problems were addressed. Each such "chunk" of information is indexed by the physical or functional part of the building it is associated with and the design issues it addresses. This case library contains cases about the design of several existing libraries and courthouses in the Atlanta area. Its database currently consists of 6 library designs, 3 courthouse designs, 67 problems, 187 stories, and 138 responses. Each story in this library discusses some problem that arose in a building design, the way that the problem was addressed, and the outcomes that resulted. Archie-2 was used once in a graduate-level design studio in the College of Architecture. A graduate level class on human-computer interface design at the College of Computing conducted a formal evaluation of the usability of Archie-2's interface. We are now in the process of extending the library by adding cases of tall building designs and building modifications for handicapped accessibility. This extended library will be tested in a graduate-level architectural design studio in the 1995-96 academic year.

Susie is a library of cases about pollution, the natural environment and industrial accidents that highlight issues of sustainable technology and development. It illustrates a different use of case libraries in classrooms: learning, not by perusing an existing library, but instead by creating one. Graduate students in a course on case-based reasoning in the College of Computing built Susie in the Winter Quarter of 1995. It holds 11 large cases, with 30 problems, 41 stories and 28 responses. Its individual stories teach lessons about making technology and development decisions that take into account environmental and ethical concerns as well as the more traditional technological and economic issues. Figure 1 shows the case presentation window of Susie with a story and accompanying problem and response statements about the Three Mile Island accident.

3. Lessons from Classroom Experiences

3.1. Broadening the Content and Presentation of Information

How can cases be organized and presented in ways that make it easy to learn from them?
DesignMuse-based case libraries provide one answer to this question in terms of individual information chunks - design overviews, problems, stories and responses. Another, for example, is the "multimedia book" style of engineering design cases developed at the University of California, Berkeley [4]. The majority of information contained in our case libraries are specific instances -
e.g., specific designs and stories. The only relatively abstract information provided are problem and response statements which generalize the lessons to be drawn from specific stories. During the course of extending Archie-2 for use in a tall building design studio, the faculty member in charge of this studio felt that having the case library also present introductory tutorial material as well as specific cases of existing tall buildings would make it more useful to students. This studio consists of two phases: introducing students to the methods, technologies, systems and issues of tall building design, followed by a design project. If tutorial materials were to be incorporated into Archie-2, it was felt that students could use the case library in both phases instead of only during the design project. Work is therefore currently underway to collect and structure introductory materials in the form of problems, stories, responses and designs connected to a “generic tall building case” in the library. In this case, explanatory materials that further illustrate a problem-response pair will replace actual stories, and generic drawings and design guidelines for a building will replace the specific floor plans or montages that typically appear in the design pane (the lowest pane in Figure 1) of the case presentation window. Another extension to Archie-2 in progress is the addition of designs for handicapped accessibility as mandated by the Americans with Disabilities Act (ADA). The case information being added is about redesign of existing buildings to conform to ADA regulations, not new building designs. This has prompted us to take a broader view of the design pane of the case presentation window and to use it to present information such as redesigns of building parts like entry ports and guidelines or regulations that governed the redesigns.

While students were constructing Susie, the one aspect they had the most difficulty with was the extraction of relevant information from the resource materials on sustainable development and technology they were provided with (e.g., newspaper articles, research papers, project reports, reports issued by agencies such as the EPA and Greenpeace) and structuring that information in the form of “chunks” that DesignMuse-based case libraries require - problems, stories, responses and designs. While problems, responses and illustrative stories lend themselves naturally to describing almost any situation, design information is not equally applicable or relevant. For example, one sustainable development case dealt with how to manage forested patches of land called “windbreaks” to prevent soil erosion as well as sustain game for sports hunters. This case has no design component to it; its focus is environmental management. Therefore it was decided to change the design pane to an overview pane that may contain any kind of contextual information - this could be an overview, a layout diagram, a time-line or similar graphical/textual device that provides sufficient context for the problems, stories and responses associated with a case. Thus, Susie displays pie-charts, tables and time-lines in this pane instead of the floor plans and montages of Archie-2.

3.2. Indexing: Vocabulary, Organization and Presentation

Case libraries use a structured indexing vocabulary for tagging information, usually created based on an expert’s understanding of the domain. Users (e.g., students) use the indexes for two purposes - to construct a probe or query during database search and to construct a set of descriptive indexes for new cases during library construction. We have found that students are not always familiar with the necessarily technical terms appearing in the indexing vocabulary. This impedes their ability to conduct useful searches and to index new information appropriately. This is motivating the addition of a glossary to the library.

Students’ efforts in constructing Susie highlighted a need to broaden the indexing system. DesignMuse was originally built to author “design” case libraries. Therefore it was felt that the indexing system of these case libraries should reflect the salient dimensions of artifact design - where (physical and functional components), what (design issues), when (lifecycle of the artifact) and who (stakeholders). Accordingly, the indexing system classifies all index terms into these classes and organizes them in a hierarchical structure for efficient searching. However, cases of sustainable development and technology required a different cut on indexes. These cases needed to be characterized in terms of the type of a case, its spatio-temporal scope, the nature of problems
encountered, the nature of possible solutions, affected parties, lifecycle stages (for technology related cases) and relevant issues. What we have discovered is that as case libraries expand into domains that go beyond design, the organizing structure and vocabulary of the indexing system can change drastically. So flexibility to change the structure of the indexing system as well as the vocabulary ought to be built into the case libraries. On the other hand, giving complete freedom to determine this structure to case library designers may result in the indexes reflecting the personal and professional idiosyncrasies of the designers which need not be apparent to the users. This will diminish the utility of the indexes. This is a tradeoff, the implications of which are yet to be investigated in classroom settings.

At present, the system presents its indexing vocabulary to users as a series of nested pop-up menus that faithfully reflect its underlying hierarchical structure. We have observed that the students get confused by the organization of the hierarchy because they are not familiar with the search and matching algorithms that exploit this hierarchical structure. In fact, students do not need to see this structure at all. Besides, the multiple levels of nested menus that this hierarchical structure engenders result in a cluttered set of pop-up menus that students find difficult to maneuver in. The solution we are implementing is to hide the hierarchy (but leave an option to see it if someone wants to) and instead to present the indexes in a more familiar form such as the indexes in a book.

3.3. Software Integration

Educational tools like case libraries can be integrated into classroom practice by either making them available for optional use by students or making their use mandatory in assignments. Our experience has been that either of these strategies does not fully engage the students. In the former, many students do not attempt to use the system. In the latter, many students use the system to the minimum extent necessary to get the assignments done. In either case, students do not exploit the full potential of the case libraries to facilitate their learning. We believe that in order to motivate and engage the students so that they derive the maximum benefit from case libraries, it is necessary to integrate their use with the rest of classroom activities. We are currently pursuing this idea in two ways. One is to integrate case libraries with software that supports design problem solving in other ways. In one project we are building an interactive problem solving system that the students will use to solve design problems, and integrating a case library with it so that one kind of feedback the problem solver will give students is to bring up the case library with a set of cases relevant to the issue they are pursuing at the time feedback was requested. This way students get started off with a set of interesting cases relevant to their problem, which should motivate them to search and explore further in the library. Another project in the College of Architecture seeks to integrate a sketch-pad with case libraries so that an architect could make sketches to guide the search for relevant cases and annotate the retrieved information. A broader approach to integration is to embed case libraries in a larger computer-based learning and work environment with many other components. Integrating a case library into an electronic workspace that provides tools for communication, collaborative work, simulation, etc. as well, for example, will facilitate the use of cases for multiple purposes. Once relevant cases have been found by searching the case library, that information can be immediately be imported into the workspace and used to solve some aspect of the design problem at hand. Cases can also be imported into collaborative discourse or argumentation. For example, cases provide one type of information a student might point to as justification for some argument, as a potential alternative to a design decision, or as a rebuttal to someone else's design decision.

3.4. Curriculum Integration

There is ongoing work in terms of further curricular use of case libraries. Archie-2, the architectural case library, is currently being extended to include cases about handicapped accessibility and tall building designs. This extended library will be fielded in a graduate-level design studio in the College of Architecture in Fall 1995. Work on the creation of a new case library consisting of product design cases has just begun. This library will be used in early 1996 in a collaborative
product development course being offered jointly by the Industrial Design Program and School of Management at Georgia Tech. Another case library documenting Olympics construction projects in the Tech campus, for use in a design course to be offered jointly by Architecture and Civil Engineering in 1996, is also underway. Finally, in the Spring of 1995, five groups of students in a sustainable technology course constructed five case studies on paper: disinfection of waste water, pulp fiction, the role of PVC, pharmaceuticals with chlorine, and sustainability of chlorine use with refrigeration. We plan to use these cases as seed materials for students to extend Susie during the next offering of this course.

4. Case Libraries in Design Education

How can students learn from case libraries? Cases, being rich knowledge structures that explicate both conceptual and strategic knowledge, will allow students to master concepts, principles and strategies in the course of attempting to solve problems. This will promote transfer. As cases will be connected to domain principles, learners understand how knowledge is applied to problems, and this should in turn lead to the acquisition of flexible knowledge. Students will analyze multiple cases from the case library and reflect on how these cases are similar and different to the problems they are solving. Case libraries also facilitate the acquisition of prior examples to apply in later situations. There are two different and equally important ways in which students will use case libraries. One is by searching for, analyzing, comparing and contrasting cases that are similar to the problems they need to solve. The other is by constructing cases from domain knowledge and from their problem solving experiences and incorporating these into the case libraries.

Use of case libraries can also scaffold skills of resource identification and use. The goal of scaffolding is to help students to carry out a reasoning process or achieve a goal that they would not be able to do without help, and to facilitate learning to achieve the goal without support [1]. Case libraries support student exploration by providing multiple ways of finding and navigating among cases. Students might begin looking at one story of interest and explore related stories by a number of different dimensions (e.g., same subsystem being designed, similar challenges, similar solutions), or begin by browsing all the stories about the same subsystem, or one of a number of other dimensions. The skill of searching for relevant information can be further scaffolded by presenting the library index in intuitive formats and encouraging students to construct and experiment with complex search probes from the index terms.

Acknowledgments

Research reported here has been supported by ARPA under contract N00014-91-J-4092 monitored by ONR, ONR under contract N00014-92-J-1234, and by the Woodruff Foundation's support of the EduTech Institute. Research on Archie-2 has been conducted in collaboration with Craig Zimring, College of Architecture, Georgia Tech.
References


A Multiple-Case-Based Approach to Generative Environments for Learning

Cindy Hmelo, N. Hari Narayanan, Wendy C. Newstetter, and Janet L. Kolodner
Edutech Institute
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
[ceh, narayan, wendy, jlk]@cc.gatech.edu

Abstract
In this paper we present a vision of learning through solving generative problems—problems that promote open-ended inquiry and have multiple solutions. This vision stems from a novel and evolving approach to learning that we are developing at the EduTech Institute. This approach is based on the following premises: that learning is facilitated by generative problem solving, collaborative work and use of multiple cases; that learning and skill acquisition need to be, and can be, scaffolded through software; and that a computer environment which integrates a shared and structured electronic workspace with a full variety of functionalities can effectively support all of the above. We describe this approach and the architecture of the corresponding computer environment. This environment is designed to serve three critical functions: provide a shared workspace for students, facilitate inter- and intra-group collaborative work, and make available the tools and resources that students need for problem solving and learning. The software components of the environment that have already been implemented are described. Finally, issues and questions driving current research are outlined.

1. Introduction
Too often students learn bits of knowledge that they are unable to assemble and apply in productive ways. Modern theories of learning suggest that for learning to occur and to be useful, the learner needs to be actively involved in meaningful contexts (e.g., Brown, Collins, & Duguid, 1989). Situating learning in rich problems or cases is one way of accomplishing this. Case-based learning (CBL) means that, as students solve a problem or study a case, they construct knowledge at the same time (Williams, 1993). In this paper we present a multidisciplinary approach to structuring learning environments within the framework of CBL, called Multiple Case-Based Approach to Generative Environments for Learning (McBAGEL). At the EduTech Institute, we are developing McBAGEL for use in a variety of engineering and middle school courses that involve ill-structured problem-solving. Several factors characterize our approach: (1) the use of design problems to promote generative learning; (2) using design cases provided by computer-based case libraries as resources to aid problem solving and foster deep understanding; (3) scaffolding the acquisition of knowledge and skills through a combination of software support and careful orchestration of classroom activities; (4) providing feedback in a variety of forms; (5) emphasizing group-oriented collaborative work in and out of the classroom; (6) encouraging reflective articulation; (7) making available to students an electronic sharable workspace integrated with tools and resources as the medium for work; and (8) redefining the role of the teacher as a facilitator of learning. Because we are dealing with design-oriented problems, there is a great deal of complexity to be managed. The pedagogy and software support is geared towards supporting learners in managing the complexity and emphasizing learning as well as doing.

McBAGEL comprises both a pedagogical approach and a software architecture designed to complement and support effective classroom practice of this approach. We are engaged in implementing this approach both at the college level (engineering undergraduates) and at the middle school level. We are currently developing design problems and case libraries for engineering design and middle school math and science curricula, as well as prototyping the computer-based electronic workspace. In this paper, we discuss the theoretical foundations of our approach, the architecture of the electronic workspace, and its software components.

2. Foundations of Case-Based Learning: A Multidisciplinary Perspective
Case-based learning (CBL) uses cases or problems as stimuli for learning and foci for organizing what has been learned (Barrows, 1994). This means that students learn while solving a problem or studying a solved case. Characteristics of CBL environments include:

• Emphasis on collaborative student-centered learning.
• Emphasis on learning knowledge and strategies.
• Students solve the problem and make their thinking visible by justifying their solution process using theory, causal models, or other appropriate evidence.
• Problems are ill-structured, the data are embedded in the problem itself, and are often emergent

Presented at 2nd Annual Symposium on Cognition & Education, Varanasi, India, December, 1995
as the problem is explored.

McBAGEL, our approach to CBL, draws from research in Cognitive Psychology, Education, and Case-based Reasoning. Relevant perspectives from these areas are discussed below.

**Cognitive Psychology.** There is both cognitive theory and evidence supporting the benefits of case-based approaches to learning (Norman & Schmidt, 1992). The cognitive psychology literature provides support for CBL in three roles. First, the acquisition of factual knowledge in the context in which it will later be used should enhance later retrieval (Adams, et al., 1988; Perfetto et al., 1983). Second, the mastery of concepts, principles, and strategies while attempting to solve a problem should promote transfer to new problems (Catrambone & Holyoak, 1989; Needham & Begg, 1991). Needham and Begg (1991) have demonstrated the importance of feedback in facilitating transfer to new problems, suggesting that this should be an important feature of CBL. Third, the acquisition of prior examples that can be used for problem solution on the basis of similarity is valuable. However, for exposure to earlier problems to be helpful the learner needs to see multiple examples and reflect on them (Brooks et al., 1991; Chi et al., 1989; Gick & Holyoak, 1983). Moreover, the collaborative discussion that occurs in CBL is important for student learning because it activates prior knowledge, thus facilitating the processing of new information (Bransford & Johnson, 1972; Dooling & Lachman, 1971; Schmidt et al., 1989). In CBL students are encouraged to think about problems with the underlying principles in mind rather than just collecting sets of features. Chi et al. (1989) note that during self-explanations cases get connected to domain principles and the learner begins to understand how knowledge can be applied to solving problems, leading to more flexible knowledge.

Cognitive Flexibility Theory (CFT) provides evidence that suggests case-based learning methods are important for learning in complex, ill-structured domains (Spiro, Feltovich, Coulson, & Anderson, 1988). A domain is ill-structured when cases are multidimensional and irregularly related to other cases and to the underlying causal models (if such models exist). By this definition, design is ill-structured. This means that, although the underlying science and mathematics used in a design problem may be well-structured, "there is variability from case to case regarding which conceptual elements are relevant and in what pattern of combination" (Spiro et al., 1988, p.379). CFT also suggests that knowledge will be learned flexibly if students have the opportunity to explore the problem from multiple points of view. The implication of this is that for cases and concepts to be understood and useful as problem-solving tools, they need to be visited from a variety of different experiences.

**Education.** Research in education provides us with two examples of case-based learning (e.g., Barrows, 1985, 1994; CTGV, 1990, 1993). Anchored instruction situates learning and problem-solving in rich contexts with meaningful goals. Students solve complex, realistic mathematics problems using a video-based story to present the problem. For example, The Adventures of Jasper Woodbury (CTGV, 1990; 1993) video series depict realistic situations in which mathematical problem solving is required. To solve these problems, the students must formulate their problem-solving goals and plan how they will achieve those goals. They must identify the information relevant to solving the problem as is required in complex real-world problem-solving. Students initially brainstorm to determine the goals and data that would be needed to solve the problem. Then, students break into small groups to work on the subproblems or subgoals that they generate. Groups present the results of their problem solving to the class and get feedback from the class and teacher. While working in small groups, students can review the video to gather data and ask the teacher questions. They learn about mathematics in the course of solving the problem.

Anchored instruction affords opportunities for problem finding, exploration, and discovery (CTGV, 1990; 1993). The stories provide for generative learning because the learners must complete the story through their problem-solving efforts. An important feature of the stories is that the data are embedded within them. The problem-solvers must learn to differentiate between relevant and irrelevant data. The problems used are much more complex than, for example, typical math word problems on the premise that children cannot learn to deal with complexity unless they have had the chance to experience it.

Problem-based learning (PBL) is a CBL methodology developed originally for medical students (Barrows, 1985). Students in PBL learn through solving real patient problems. The PBL approach begins with the teacher presenting a problem to students by showing materials or facilitating a discussion that "brings the problem home" to students. The students are presented with an initial scenario that specifies a problem to be solved and the particular product or performance that they must achieve. As they try to better understand the problem, they question the facilitator to get additional case information. At several points in their problem solving, the students pause to reflect on the data they have collected so far, to generate questions about the data, and to hypothesize about underlying causal mechanisms or solutions for the problems. Students identify issues that they do not understand and need to learn more about. After considering the problem with their naive knowledge, students research the learning issues they have identified, by going to the library, consulting an expert, interacting with a computer program, or in whatever way is appropriate. They later share what they’ve learned, attempting to apply their new knowledge in solving the problem. They might refine hypotheses or options or generate new ones based on their new knowledge. The cycle of deliberating together and separating to find out new things continues until the problem is solved. When finished, the students present their product and then intentionally reflect on what they have learned. In addition, they assess their own and other group members’ contributions to the group’s learning and collaboration.

PBL has several goals. First, in order to make basic science knowledge available in a clinical context, students learn science
by having it embedded in a patient problem. This should help students better integrate scientific and clinical knowledge, thereby improving their access to basic science information when they need it in the clinical context. A second goal is to facilitate the development of clinical reasoning skills. A third goal is to facilitate the development of skills in self-directed learning and self-assessment. An explicit objective of PBL is to increase students’ sensitivity to their personal learning needs and their skill at locating and using appropriate information resources. Finally, PBL is expected to enhance student motivation. PBL provides an appropriate context for learning because all of the content is learned in the context of a problem which should enhance recall of this information when it is needed. Norman and Schmidt (1992) report several direct tests of the effect of PBL on recall. These studies indicate that PBL students’ initial learning is not as good as students in a conventional curriculum but that their long term retention is superior. Research has shown that students in a PBL environment are more likely to use science in their problem-solving than traditional students (Hmelo, 1994).

Case-Based Reasoning. Case-Based Reasoning (CBR) provides insights that complement the CBL methodologies. It provides suggestions for sequencing problems to form a curriculum as well as the kind of reflection that is needed (Kolodner, 1995). CBR means reasoning based on previous experiences (Schank, 1982). It might mean solving a new problem by adapting an old solution or merging pieces of several old solutions, interpreting a new situation in light of old similar situations, or projecting the effects of a new situation by examining the effects of a similar old situation. In short, case-based reasoning means using the lessons learned in old situations to understand or navigate new ones. The basic premise underlying case-based reasoning is the preference to reason using the most specific and most cohesive applicable knowledge available. Inferences made using cohesive knowledge structures, i.e., those that tie together several aspects of a situation, are relatively efficient. Cases, which describe situations, are both specific and cohesive. In addition, they record what is possible, providing a reasoner with more probability of moving forward in a workable way than is provided by knowledge that is merely plausible.

Research in CBR (Kolodner 1993) indicates several ways of enhancing CBL. It suggests that knowledge will be more accessible, flexible, deeply learned, and accurate if learners have the opportunity to encounter (first-hand or by report) multiple situations in which the knowledge is used and multiple ways in which similar situations are addressed, and if students have the opportunity to reuse and try out knowledge gained through experience. Further, CBR suggests that the experiences of others be made available to students to model successful reasoning, to help students get started, to point the way to issues that need to be addressed, and to fill in where the full range of real experience is impossible or infeasible. It also suggests that problems and products should afford failure of expectations and that the environment needs to afford the kinds of feedback that will allow successful analysis of failures. Finally, it suggests that reflection should focus on anticipating the uses of lessons learned through each experience and that facilitation should refer back to previous experiences of the students to help them notice similarities and abstract from the range of problems they’ve solved.

3. Synthesis

When we put all that we have learned from research in cognitive psychology, education, and case-based reasoning together, four principles emerge:

1. **Learning is enhanced by generative problem solving.** Learning is most effective when it occurs through generative activities associated with solving problems (e.g., identifying and formulating the problem, generating alternatives, evaluating, decision making, reflecting, and articulating). Design, by its very nature, is a generative activity. Therefore, design-oriented problems are particularly effective for technical domains like engineering as well as providing effective contexts for math and science learning.

2. **Collaborative work promotes knowledge building.** Students are expected to solve problems and do assignments in groups. Group-oriented work, in and out of the classroom, is important both in facilitating learning and in preparing students for today’s multidisciplinary team-oriented workplaces. As students work in collaborative groups, they are forced to articulate and reflect upon their thinking, leading to an appreciation of the importance of distributed cognition (Pea, 1993) as well as enhancing learning and subsequent transfer (Brown & Palincsar, 1989). Collaborative work allows students to successfully tackle problems more complex than what any one group member could do alone. The collaborative discussion that occurs is important for student learning because it activates prior knowledge, thus facilitating the processing of new information (Bransford & Johnson, 1972; Schmidt et al., 1989). The dialogue among group members enriches and broadens the groups’ problem solving process (Vye, Goldman, Means, Voss, Hmelo, & Williams, 1995). On the other hand, Blumenfeld et al. (1991) suggest that students may have more motivation to learn but make less use of learning and metacognitive strategies. In addition students may not have the skills to benefit from collaborative work. Therefore it is important to help students to collaborate well together in order to make collaborative learning work well.

3. **Articulation and reflection are central to learning.** Several researchers have demonstrated the importance of articulation and reflection to learning. The goal of reflection is to analyze and evaluate one’s knowledge, learning and problem solving strategies. Priolli and Recker (1994) suggest that reflection on problem solutions that focuses on understanding the abstract relationships between problems is related to improved learning. Lin (1994) has found that reflection on problem-solving processes leads to enhanced transfer and that technology can be used to scaffold appropriate kinds of reflections. One way that
reflection can be enhanced is through the articulation of metacognitive knowledge and skills that typically occurs in collaborative discourse.

4. Access to multiple cases will facilitate flexible learning. Providing students with access to multiple cases that contain rich descriptions of specific situations can significantly enhance learning and transfer. The use of multiple cases as resources for learners’ problem solving both facilitates learning new knowledge, and supports the skill of transferring previous solutions to the current problem. It is expected that by revisiting concepts and skills through numerous cases, flexible transfer will be promoted (Spiro, Coulson, Feltovich, & Anderson, 1988). Consideration of a single case leads to inflexibility of the acquired knowledge and strategies (Williams, Bransford, Vye, Goldman, & Carlson, 1993). Therefore, it is important to provide students with access to multiple cases that are relevant to their activities during various stages of design problem solving. Computer-based case libraries can provide students with not only such access but also means of flexibly navigating among cases and parts of cases. Such libraries are therefore information-rich and powerful resources for learning. Students will use case libraries in two significant ways. One is by searching for, analyzing, comparing and contrasting cases that are similar to the problems they are solving. The other is by constructing cases from domain knowledge and from their problem solving experiences and incorporating these into the case libraries.

4. McBAGEL

Our approach, called McBAGEL, derives from these principles. McBAGEL proposes to situate classroom learning in information-rich contexts that afford opportunities for problem formulating, exploration, and discovery. Students work on problems for extended periods of time, reflecting and articulating on both the process and the product. Case libraries provide them with both relevant data and specific solution strategies in the domain of instruction, all within the context of complex and realistic problems. The problems students have to solve and the cases that are made available to them serve as anchors for learning. Collaborative, reflective and articulative activities, aided by the tools and cases provided by the computer-based learning environment, are expected to improve the students’ knowledge, problem solving skills, and self-directed learning skills. Cases, being rich knowledge structures that explicate both conceptual and strategic knowledge, will allow the students to master concepts, principles and strategies in the course of attempting to solve problems. Students will revisit ideas from multiple cases both through the sequence of design problems that they work on and the design cases they access in the case libraries. We believe that by having students analyze multiple cases, and by having them reflect on how these cases are similar and different from the problems they are solving, more flexible knowledge should be constructed. Cognitive flexibility theory supports this prediction (Spiro et al., 1988). The collaborative nature of student activities should facilitate the construction of new knowledge since it encourages articulation and intra-group communication.

We take a holistic approach that includes development of curriculum-appropriate design problems, scaffolding the complexity, software development and teacher involvement. Our approach is distinguished by the following features:

1. Scaffolding. Solving real-world problems is hard. Others' experiences on implementing effective problem-based learning environments teach us that solving complex problems requires scaffolding, i.e., help from facilitators, knowledgeable experts, and the learning environment, to help students manage the complexity of problem-solving and to promote learning (Koschman, Myers, Feltovich, & Barrows, 1994; Scardamalia, Bereiter, McLean, & Woodruff, 1989). The goals of scaffolding are to enable students to carry out a reasoning process or achieve a goal that they would not be able to do without help, and to facilitate learning to achieve the goal without support (Brown, Collins, & Duguid, 1989). In McBAGEL, explicit scaffolding of different skills will be provided by expert teachers, through software, and by orchestrating appropriate uses of multimedia tools such as collaboration software, simulation and visualization programs, and decision-support systems. Use of case libraries can also scaffold skills of resource identification and use. Case libraries support student exploration by providing multiple ways of finding and navigating among cases. The skill of searching for relevant information will be scaffolded by presenting the library index in intuitive formats, and encouraging students to explore the library by constructing complex search queries with multiple index terms and conducting searches.

2. Feedback. Providing appropriate feedback is important. One of the key features of problem-solving in the real world is that the problem-solver takes some action and receives feedback from the real world. Research in CBR and cognitive psychology also point out the importance of feedback in coaching (Lesgold, 1994) and in learning from experience. Individuals use feedback to judge how effective their problem-solving efforts were. Feedback can take many forms—for example, it may come from the results of an experiment, the success (or failure) of building and testing working models, or modeling a process by constructing and running a simulation. Previously solved cases may be used to provide anticipatory feedback such as warning the problem solver of a potential problem with a solution that was tried previously. McBAGEL suggests providing students with feedback in a variety of forms and range of fidelity—feedback from teachers, feedback from peer groups, feedback from prior experiences encoded as cases, evaluations by experts on-site or on-line, feedback from computer simulations, and feedback from the real world when working models are built and tested.

3. Emphasis on design problems. Several aspects of design problems make them ideal for promoting learning. (1) They are generally under-specified but might have some over-constrained parts, making understanding and problem definition crucial to
the solution. (2) They have specified clients or audiences who must be satisfied, making the relevance and human dimension of the problem clear. (3) Problems in design are not operational in that a clear-cut path through the problem space is not generally available. Multiple routes to differing destinations exist for the student and the way through is something they must address and navigate. Successful solution requires exploration, questioning, and evaluation. (4) Successful design is iterative. Several to many alternatives are attempted, partial solutions are explored, dry-runs and evaluation occur in the process until several rounds result in a movement forward toward a solution. Students who tend to be task- or product-driven can benefit greatly from participating in such a process. (5) In moving through a design space, numerous criteria have to be managed simultaneously. In any manufacturing or construction problem, for example, cost, manufacturability, availability of materials, time, and environmental issues co-occur as constraints to be tackled. Managing this complexity mirrors what students often know about problems in the real world, lending a sense of authenticity to the tasks while distinguishing design problems from back of the book problem solving. (6) Real-world design problems often have several parts that interact in interesting ways and that require a variety of different kinds of domain knowledge to be solved. When parts of problems interact, there is usually no one right way to address the interaction; rather, successful solution depends on considering the many ways of trading off interactions against each other and choosing between alternatives. (7) The need for interaction with the real world makes design problems nice as well; design problems are situated in the world we live in, making their relevance and the relevance of the math and science needed to solve them clear. At the same time, making a solution operational requires interaction with the real world and thus affords feedback.

4. **Collaboration.** Collaboration is a key piece of our pedagogy. Research on collaborative learning shows that learning while solving problems in groups facilitates the learning of articulation skills, makes learning more effective for all group members, and allows students to successfully tackle problems more complex than any one group member could individually solve (Brown & Palincsar, 1989; Pea, 1993; Vye et al., 1995). Aspects of our approach - the division of the student body into small groups, the complexity of the design problems that the groups will tackle, and the use of collaboration software to scaffold communication and cooperative work - are all intended to overcome these limitations and enhance the benefits of group-oriented learning. In our approach, scaffolding will be provided to help students learn to collaborate as well as learn through collaboration.¹

5. **Reflective articulation.** An important aspect of our approach is promoting reflective articulation - articulation that engenders reflection, leading to enhanced understanding. There are several forms of reflective articulation including generating analogies, predicting outcomes of events or processes, developing questions about the learning materials, and self-explanations (Chi et al., 1989; Weinstein & Mayer, 1985). Reflective articulation can enhance retention, elucidate the coherence of current understanding of the problem being solved, improve self-directed learning skills, and provide a mechanism for abstracting knowledge from the content in which it was learned, thus facilitating transfer. McBAGEL emphasizes two levels of reflective articulation - individual and group - within a collaborative learning environment.

6. **A shared electronic workspace.** We believe that an electronic workspace which seamlessly integrates a full variety of functionalities, tying together tools that students will use for collaboration, communication and problem solving, will significantly enhance learning. Such an environment is an ideal vehicle for providing adaptive software-realized scaffolding of various skills. The functionalities that this workspace will provide include: an electronic notebook with both private and sharable sections; case libraries and other information resources; tools for simulation, visualization, etc.; a tool for communication, collaboration, and multimedia document sharing; and a set of basic tools such as word processing programs and spreadsheets. Such an integrated computer-based learning environment that students use as a "professional workspace" is a central component of McBAGEL. It is not sufficient to confine such an environment to a laboratory or classroom. Instead, it needs to be made available to students across courses and across campus (e.g., available in all public computer labs) for providing easy access at all times as well as continuity across the curriculum. This availability should encourage both synchronous and asynchronous collaborative work among students.

7. **The centrality of teachers.** Helping teachers learn to become expert facilitators and partners in the development process is critical (CTGV, 1993). With student-centered learning, the role of the teacher is increasingly important in facilitating student learning and orchestrating classroom activities. The commitment of teachers to student-centered learning is crucial (e.g., Barrows, 1994). Teachers have many roles in our approach - as expert facilitators of problem solving, learning and collaboration; as full partners in the orchestration of classroom activities; as integrators ensuring that the use of computers and software tools is naturally integrated with other activities; and as cognitive diagnosticians.

There are many challenges in achieving the goals we have described. Students do not necessarily view issues from multiple perspectives nor do they collaborate well. They will often recall rather than reflect. An expert facilitator can help promote these processes but often large class size precludes the small group work that would afford these experiences. We believe that providing computational support - making the right kinds of software and on-line information available for use at the right times and seamlessly integrating the use of computers for communication and problem solving - can alleviate some of these difficulties.
5. Computer Support for Learning in McBAGEL

The software environment for McBAGEL (Narayanan, et al., 1995) needs to address several key requirements: access to information resources such as case libraries; support for synchronous and asynchronous collaboration; support for reflective articulation; and provide tools to support problem solving. Figure 1 is a schematic diagram of its architecture. This environment provides an external memory for keeping track of problem specifications, important facts and constraints, ideas about how to deal with the specifications, and learning requirements. The main screen provides several fields for keeping track of multiple sources of information, design alternatives, and further actions to be taken. Space is provided to record the facts and constraints that are important, to record ideas about how to deal with the specifications, and to keep track of what else needs to be learned, what information needs to be collected, and what actions need to be taken. Together, these windows allow the student to see where she is now, where she has been, and where she is going. This screen can be used as an individual workspace or as a shared workspace for the group. The main screen (Figure 2) also provides access to other resources and tools that students need to solve the design problems: case libraries and other information resources; tools for simulation, visualization, decision making etc.; a tool for inter- and intra-group communication, collaboration, and multimedia document sharing; and a set of basic tools such as document processing programs, drawing/painting programs and spreadsheets.

The problem screen (Figure 3) provides easy access to the evolving problem description. This screen begins with a minimal description of the design problem presented to the students. Details emerge as they inquire about additional information on constraints, material resources and functional issues regarding the design. The collaboration window allows students to enter into a collaboration environment that provides much more than mere communication facilities. It will provide an ability to enter into structured discussions on different topics pertaining to the class and the problem at hand as well as to share multimedia resources with other members of the group and class. A user will be able to browse through past and ongoing discussions which are presented in a structured format to allow easy topic-based, time-based or author-based browsing, and to contribute to those discussions by constructing and sending different types of messages. This collaboration facility will be made available not only to students, but also to teachers. It will provide teachers with a means to collaborate in conducting a course and to share experiences and learn from each other. It can also be a vehicle for student assessment based on their collaborative interactions.

In addition to providing a work environment, this system makes available scaffolding to help novices with design, collaboration, and reflection. Design scaffolding will vary as a function of the design stage students are working on. For example when the students are working on problem formulation, the software will provide coaching to help them understand what is involved in this stage: e.g., identifying the problem, formulating the problem, partitioning/decomposing the problem, and framing the problem. The collaboration software will provide procedural facilitation to aid in the development of collaboration skills. Reflection will be facilitated through the articulation that occurs during collaborative problem solving and learning activities.

In summary, this environment will provide ways to organize and manage projects from the students’ perspective (e.g., the main screen provides for explicitly listing organizational and learning issues) and the teachers’ perspective (e.g., tracking student progress and keeping records of student work). In addition, we envision that the environment will be used for research purposes (e.g., archiving data on inter- and intra-group communications and resource sharing for later assessment, collecting data to be used for student/group modeling in order to devise better course- and student-specific on-line scaffolding and coaching methods, etc.). An initial prototype of this environment has been developed with Hypercard on the Macintosh platform, but it has not yet been tested in a classroom. Borrowing from the metaphor of the white board workspace of problem-based learning found in medical schools, this prototype provides an electronic workspace that is split into four regions. It also allows easy access to other tools and resources. Figures 2 and 3 show the workspace and problem screens of this prototype.

Here is a brief scenario to illustrate how we imagine the students will use this environment. Students, who will be working in small groups, enter the environment at the main screen (Figure 2), which represents their shared electronic workspace. They are provided with relevant information on the design problem they need to solve via the button “problem information”. In this case, it is to design an archery stadium for the Olympics. As students are initially formulating and understanding the problem, they will be encouraged to identify data relevant to the problem from the information they have been provided with.
articulate this by recording those in the "facts" space. Similarly, as they consider alternative solutions, they will make use of the "ideas" space. The problem-based learning methodology that this environment embodies explicitly prepares students for self-directed learning by requiring them to identify their knowledge deficiencies in the "need to learn" space and the actions they plan to take to remedy those deficiencies in the "action plan" space. Several buttons are found on the bottom of the screen that provide access to different tools that they will need to solve the problem. "Stage" is a pull-down menu which acts as a gateway to various kinds of software-realized scaffolding tailored to different stages of problem solving (Guzdial, 1994).

While the structure of this environment is still evolving, some of its components have already been designed, implemented and individually fielded in classrooms. In the remaining two sections we elaborate on these implemented components and describe future directions for our research.

6. Implemented Components

- **Case Libraries**: Case libraries organize cases in ways that make it easy to access their most interesting parts, understand their implications, and recognize the range of problems needing solving and the range of solution methods available. Case studies are structured in terms of overviews, problems, stories, and responses. Each story discusses some problem that arose in designing some artifact, the way that the problem was addressed, and the outcomes that resulted. To make it easy for users to extract from stories their important points, stories are presented with illustrative graphics, and several kinds of contextual information is associated with each story. Students can examine the full artifact that some story is associated with, can see a general description of the problem the story addresses, a general description of the kind of solution it provides, and can ask to follow links to other stories that illustrate a similar problem or solution. The stories help students discover which issues they should be considering during design and help them to anticipate the results of carrying out their proposed designs. We have developed a number of case libraries in support of design problem solving.

- **Case Library Authoring Tool**: DesignMUSE (Domeshek & Kolodner, 1992) is a case library authoring tool that has been developed to allow easy construction of case libraries. Student-faculty teams have used it to create case libraries in the domains of architectural design and sustainable technology. While existing case libraries act as information resources, this authoring tool will allow students to construct their own case libraries to record the design problems that they solve.

- **Collaboration support**: The McBAGEL screens themselves provide support for synchronous collaboration by giving the students a shared context for discussion. Our asynchronous collaboration software CaMILE (Guzdial, Rappin, & Carlson, 1995), based in principle on CSILE (Scardamalia et al., 1989), integrates information-gathering tools, communication tools, and applications into a collaborative environment. CaMILE provides a discussion environment into which the full range of text, graphics, spreadsheets, video, and so on that reside locally or on the Internet can be incorporated. It is designed to meet two goals. First, it serves as a collaboration and information indexing tool. Discussions are structured and annotated with links to material anywhere across the network. Second, it serves as a design support tool. Discussions about design problems can be annotated with links to actual ongoing designs. The discussion trace can then serve as a design rationale and a case study of a design. It allows students to collaborate in learning and problem solving by providing a facility for structured inter-group and intra-group communications that are archived, and by providing a way to share multimedia documents easily among collaborators. Like CSILE, CaMILE scaffolds collaboration through procedural facilitation. While electronic mail merely allows team members to share ideas, CaMILE helps them to organize their ideas into coherent arguments, relate their ideas to one another, and use resources across the network to support their arguments. CaMILE can be accessed via the World Wide Web.

- **Exploratory Simulations**: We are developing a range of exploratory simulations (e.g., Rappin, Guzdial, Ludovice, & Reaiff, 1995) that enable students to learn through simulated experience. Key to these simulations are tight integration with real world problems that the students will be solving, and flexible specification of simulation choices to allow for creative and sophisticated simulation problem solution with immediate feedback. Simulations linked to the middle school science and math problems we are developing are being built with applications like the Logo Microworlds on Macintosh computers.
7. Promises, Pitfalls and Research Directions

Our approach facilitates the acquisition of factual knowledge in the contexts in which it is likely to be used later in workplaces. This is accomplished by both the use of real-world problems in classroom activities and the provision of multiple real-world cases. Cases, being rich knowledge structures that explicate both conceptual and strategic knowledge, will allow the students to master concepts, principles and strategies in the course of attempting to solve problems. This should promote transfer. As cases will be connected to domain principles, the learner will be able to understand how knowledge is applied to problems, and this should in turn lead to the acquisition of flexible knowledge. Case libraries also facilitate the acquisition of prior examples to apply in later situations. The collaborative nature of student activities should facilitate the assimilation of new knowledge since it encourages articulation and intra-group communication. All these factors should together also result in longer term retention, and successful application of the learned knowledge beyond the classroom.

The major pitfall that has been identified in the use of CBL is the inflexibility of the knowledge and strategies acquired (Williams, et. al., 1993). In the anchored instruction environment, when students were asked to solve a related problem, they often used the original solutions and did not appropriately adapt it to the new situation. When students were asked to predict the solution to “what-if” variants of the problem and received feedback via a simulation of the problem situation, transfer became more flexible as students developed more accurate causal models of the problems (Williams, 1994). In the McBAGEL approach, rather than having students revisit a single case, we have students revisit ideas from multiple cases both through the design problems that students work on and the cases in the case libraries. Students will reflect on, and articulate, how different cases are similar and different to the problems. The problems that students solve are design problems which we believe require a different variety of reasoning strategies from the diagnostic problems traditionally used in PBL. The design cases that are made available to students to use during problem solving contain information both in the form of problem-solution pairings as well as elaborations such as causal models and theoretic justifications. All these factors ought to enable students to construct flexible and coherent knowledge structures and grasp multiple problem-solving strategies. McBAGEL also provides a long-term record of students’ problem-solving thus allowing them to revisit their earlier experiences in later efforts.

Issues of knowledge flexibility can be addressed by attending to the kinds of learning strategies encouraged in CBL. Bassok and Holyoak (1993) make a distinction between top-down and bottom-up learning. Top-down learning depends on prior knowledge of the domain coupled with active learning strategies that allow the learner to make principled judgments about the importance of features to the learner’s goals. Bottom-up learning refers to inductive learning by examples. Bottom-up learning requires that students make generalizations from multiple examples. The learners do not engage in deep analysis of principles and may end up knowing sets of correlated features (including some that are irrelevant). To the extent that top-down learning enables learners to successfully identify relevant but non-obvious features of a problem, more flexible transfer will be promoted (Bassok & Holyoak, 1993; Patel & Kaufman, 1993). Several studies have demonstrated that PBL students are likely to take a hypothesis-driven approach to their own learning and thinking (Hmelo, 1994; Hmelo, Gotterer, & Bransford 1994; Patel, Groen, & Norman, 1993). Research on learning and transfer suggests that a hypothesis-driven or top-down approach to learning may be advantageous (Chi, Bassok, Reimann, Lewis, & Glaser, 1989; Bassok & Holyoak, 1993). If the domain knowledge is fragmented however, students may need to have their attention directed to goal-relevant aspects of the problem.

The McBAGEL approach, as it is being further refined and implemented, provides a research opportunity to test our theories of learning. There are two sets of interrelated issues: those related to knowledge and strategy use and those related to the technology that we will use to scaffold learning and transfer. The issues of flexible knowledge and strategy use are critical for understanding the success of this approach. Understanding the way that learners use cases to help them in problem-solving is an important research issue that will help us develop activities that facilitate using cases as effective learning tools. There are also technology issues related to the types of scaffolding and tools provided for the students. We need to better understand how these tools are best integrated and implemented from a cognitive perspective. Some of the related questions which we plan to address as part of our ongoing research are the following. What are students learning? What are the different ways in which cases can be used most effectively in the classroom, in conjunction with student-centered problem solving activities? How can the influence of cases in promoting learning, integration, retention and transfer be accurately measured? What are the disadvantages of using cases and problems in instruction? What makes for an effective case? How can technology be used for effective and adaptive scaffolding? What kinds of student-teacher, student-student, student-computer and teacher-computer interactions should be supported?

8. Conclusion

Case-based learning environments have the potential for helping students to construct usable knowledge and to learn strategies that prepare them for a lifetime of learning. To afford generative learning, the environments need to contain rich sources of information. In addition, opportunities for student articulation and reflection must be provided to help students think deeply about the problems they are working on and to learn to go beyond the problems given. The McBAGEL approach addresses these issues by situating learning in design problems and by providing an integrated learning environment that contains case libraries, simulations, and other tools for learning. By combining the best aspects of other CBL models with new ideas, and by
drawing from research in Education, Cognitive Psychology and Case-Based Reasoning, McBAGEL represents a multi-disciplinary approach to innovation in educational practice.

Acknowledgments
Research reported here has been supported by ARPA under contract N00014-91-J-4092 monitored by the Woodruff Foundation's support of the EduTech Institute.

References


---

1 Thanks to Jennifer Turns for pointing out the importance of this distinction based on her observations of a pilot course.

2 Logo Microworlds is a registered trademark of Logo Computer Systems Inc.
Computational Support for Collaborative Learning through Generative Problem Solving

N. Hari Narayanan, Cindy E. Hmelo, Valery Petrushin, Wendy C. Newstetter, Mark Guzdial and Janet L. Kolodner

EduTech Institute, College of Computing, Georgia Institute of Technology

Abstract
In this paper we present a vision of computer-supported collaborative learning through solving generative problems - problems that promote open-ended inquiry and have multiple solutions. This vision stems from a novel and evolving approach to collaborative learning that we are developing at the EduTech Institute. This approach is based on the following premises: that learning is facilitated by generative problem solving, collaborative work and use of multiple cases; that learning and skill acquisition need to be, and can be, scaffolded through software; and that a computer environment which integrates a shared and structured electronic workspace with a full variety of functionalities can effectively support all of the above. We describe this approach and the architecture of the corresponding computer environment. This environment is designed to serve three critical functions: provide a shared workspace for students, facilitate inter- and intra-group collaborative work, and make available the tools and resources that students need for problem solving and learning. The software components of the environment that have already been implemented are described. In the final section we frame ongoing and planned research and development efforts in terms of the characteristics desired of such an environment and ways of assessing its impact.

Keywords — case-based methods of instruction, educational groupware, instructional strategies and approaches.

1. Introduction
Too often, classroom instruction provides students with many bits of knowledge that they are never able to assemble and apply in productive ways, particularly outside the classroom walls. One reason for this is the focus of traditional schooling on learning isolated facts in compartmentalized disciplines. Not surprisingly, this knowledge often cannot be transferred to real-world problems. Theories of constructivism and situated cognition suggest that for learning to be useful the learner needs to be actively involved in constructing new knowledge within meaningful contexts, not merely absorbing it. Furthermore, learning is enhanced by group-oriented collaborative work, reflection and articulation. These are therefore the central premises of a multidisciplinary approach to structuring learning within the context of case-based instruction that we are developing at the EduTech Institute. This approach is called Multiple Case-Based Approach to Generative Environments for Learning (McBAGEL).

Three factors distinguish this approach: (1) The use of generative problems to promote learning. Generative problems are those that motivate open-ended inquiry, whose solutions require synthesis, which have multiple solutions, and which, therefore, promote the generation, evaluation and combination of ideas in the course of problem solving. The type of generative problems that we use are design problems. (2) The use of multiple cases provided by computer-based case libraries as knowledge sources to aid problem solving. (3) The emphasis on software-scaffolded and group-oriented collaborative work in and out of the classroom.

We are designing a computer-based learning environment that we expect students to use as a workspace for conducting work as part of this approach. The architecture of this environment and its components is the main topic of this paper. However, since the environment's role is to support collaborative learning in the context of generative problem solving, a discussion of the approach and the educational philosophy behind it precedes the description of the computer environment. Then, software components of the environment that have already been designed, implemented and used in classrooms of Georgia Tech are described. In the final section we frame ongoing and planned research and development efforts in terms
of the characteristics desired of such an environment and ways of assessing its impact.

2. Educational Framework

Our approach is based on a synthesis of ideas on learning and problem solving from the fields of education, cognitive psychology and artificial intelligence. This approach is based on the following five central tenets.

1) Learning is enhanced by problem solving. Learning is more effective when it occurs through activities associated with solving generative problems (e.g., identifying and formulating the problem, generating alternatives, evaluating, decision making, reflecting, and articulating) rather than through transmission models of instruction. Design, by its very nature, is a generative activity. Therefore, design-oriented problems are particularly effective for technical domains like engineering and architecture and may well provide effective anchors for math and science learning.

2) Collaborative work is central to learning. Students are expected to solve problems and do assignments in groups. Group-oriented work, in and out of the classroom, is important both in facilitating learning and in preparing students for today's multidisciplinary team-oriented workplaces. As students work in collaborative groups, they are forced to articulate and reflect upon their thinking, leading to an appreciation of the importance of distributed cognition [14] as well as enhancing learning and subsequent transfer [3]. Collaborative work allows students to successfully tackle problems more complex than what any one group member could do alone.

3) Access to multiple cases will facilitate flexible learning. Providing students with access to multiple cases that contain information-rich and contextualized descriptions of specific situations set within the broader context of a course can significantly impact learning and transfer. The availability and use of multiple cases during problem solving facilitates learning new knowledge, and supports the adaptation and transfer of previous solutions to the current problem [11]. It is expected that by revisiting design skills through numerous cases, flexible transfer of these skills will be supported [20]. Intelligent computer-based case libraries can provide students with not only such access but also means of flexibly navigating among cases and parts of cases.

4) Learning and the acquisition of problemsolving skills need to be scaffolded. The experiences implementing effective problem-based learning environments teach us that solving real-world problems requires scaffolding, i.e., help from facilitators, knowledgeable experts, and the learning environment [12, 18]. The goals of scaffolding are to enable students to carry out a reasoning process or achieve a goal that they would not be able to do without help, and to facilitate learning to achieve the goal without support. The scaffolding of different skills can be provided through software, by appropriately utilizing multimedia and tools such as collaboration software, simulation and visualization programs, decision-support systems and smart case-libraries.

5) A shared electronic workspace that seamlessly integrates a full variety of functionalities for the above will enhance learning. This workspace will tie together tools that students will use while solving problems, collaborating, and pursuing multiple cases. It is also an ideal vehicle for providing adaptive software-realized scaffolding of various skills. Finally, it will encourage both synchronous and asynchronous collaborative work among students. Such an integrated yet flexible computer-based learning environment that the students use as a 'professional workspace' is a central component of our approach.

We want to situate classroom learning in information-rich contexts that afford opportunities for problem formulation, exploration and discovery. Students will work on problems for extended periods of time, reflecting and articulating on both the process and the product. Case libraries will provide them with both relevant data and specific solution strategies in the domain of instruction, all within the context of complex and realistic real-world problems. The problems students have to solve and the cases that are made available to them serve as anchors for learning. Collaborative, reflective and articulative activities, aided by the tools and cases provided by the computer-based learning environment, should improve the students' knowledge, problem solving skills, and self-directed learning skills. Cases, being rich knowledge structures that explicate both conceptual and strategic knowledge, will allow the students to master concepts, principles and strategies in the course of attempting to solve problems. The collaborative nature of student activities should facilitate the construction of new knowledge since it encourages articulation and intra-group communication. Our approach is designed in particular to address the following three issues.

Cognitive Flexibility and Transfer. Consideration of a single case leads to inflexibility of the acquired knowledge and strategies [22]. Rather than having students focus on a single case, our intention is to have students revisit ideas from multiple cases both through the design problems that students work on and the design cases in the case libraries. We believe that by having students analyze multiple cases, and by having them reflect on how these cases are similar and different to the problems they are solving, more flexible knowledge should be constructed. The cognitive flexibility theory [20] supports this prediction.

Collaboration. Collaboration is a key piece of our approach. Research on collaborative learning shows...
that learning while solving problems in groups facilitates the learning of articulation skills, makes learning more effective for all group members, and allows students to successfully tackle problems more complex than any one group member could individually solve [3, 14, 17]. Moreover, the collaborative discussion that occurs is important for student learning because it activates prior knowledge, thus facilitating the processing of new information [2, 19]. On the other hand, Blumenfeld et al. [1] suggest that students may have more motivation to learn but make less use of learning and metacognitive strategies. In addition students may not have the skills to benefit from collaborative work. Therefore it is important to help students to collaborate well together in order to make collaborative learning work well. Aspects of our approach - the division of the student body into small groups, the complexity of the design problems that the groups will tackle, and the use of collaboration software to scaffold communication and cooperative work - are all intended to overcome these limitations and enhance the benefits of group-oriented learning.

Reflective Articulation. Two important aspects of our approach are articulation and reflection. There are several forms of reflective articulation including generating analogies [10], predicting outcomes of events or processes [21], developing questions about the learning materials [10], and self-explanations [4]. Studies suggest that reflective articulation can enhance retention, elucidate the coherence of current understanding of the problem being solved, develop self-directed learning skills, and provide a mechanism for abstracting knowledge from the content in which it was learned, thus facilitating transfer. It is very important to provide two levels of articulation - individual and group - in a collaborative learning environment. Also, it is not just articulation by itself that is important, but it is the specific kinds of articulations that engenders reflection - reflective articulations - that lead to enhanced understanding. The goal of reflection is to analyze and evaluate one's knowledge, learning and problem solving strategies. Several researchers have demonstrated the importance of articulation and reflection in learning. Pirolli and Recker [15] suggest that reflection on problem solutions that focuses on understanding the abstract relationships between problems is related to improved learning. Lin [13] has found that reflection on problem-solving processes leads to enhanced transfer and that technology can be used to scaffold appropriate kinds of reflections. One way that reflection can be enhanced is through the articulation of meta-cognitive knowledge and skills that typically occurs in collaborative discourse.

3. Computer Support for Collaborative Learning in McBAGEL

Figure 1 is a schematic diagram of the architecture of a software environment that we are developing to complement the McBAGEL approach in classrooms. This environment provides an external memory for keeping track of problem specifications, important facts and constraints, ideas about how to deal with the specifications, and learning requirements. The main screen provides several fields for keeping track of multiple sources of information, design alternatives, and further actions to be taken. Space is provided to record the facts and constraints that are important, to record ideas about how to deal with the specifications, and to keep track of what else needs to be learned, what information needs to be collected, and what actions need to be taken. Together, these windows allow the student to see where s/he is now, where s/he has been, and where s/he is going. This screen can be used as an individual workspace or as a shared workspace for the group. The main screen also provides access to other resources and tools that students need to solve the design problems: case libraries and other information resources; tools for simulation, visualization, decision making etc.; a tool for inter- and intra-group communication, collaboration, and multimedia document sharing; and a set of basic tools such as document processing programs, drawing/painting programs and spreadsheets.

The problem screen provides easy access to the evolving problem description. This screen begins with a minimal description of the design problem presented to the students. Details emerge as they inquire about additional information about constraints, material resources and functional issues regarding the design. The collaboration window allows students to enter into a collaboration environment that provides much more than mere communication facilities. It will
provide an ability to enter into structured discussions on different topics pertaining to the class and the problem at hand as well as to share multimedia resources with other members of the group and class. A user will be able to browse through past and ongoing discussions which are presented in a structured format to allow easy topic-based, time-based or author-based browsing, and to contribute to those discussions by constructing and sending different types of messages. This collaboration facility will be made available to not only students, but also to teachers. It will provide teachers with a means to collaborate in conducting a course and to share experiences and learn from each other. It can also be a vehicle for student assessment based on their collaborative interactions.

In addition to providing a work environment, this system makes available scaffolding to help novices with design, collaboration and reflection. Design scaffolding will vary as a function of the design stage students are working on. For example when the students are working on problem formulation, the software will provide coaching to help them understand what is involved in this stage: e.g., identifying the problem, formulating the problem, partitioning/decomposing the problem, and framing the problem. The collaboration software will provide procedural facilitation to aid in the development of collaboration skills. Reflection will be facilitated through the articulation that occurs during collaborative problem solving and learning activities.

In summary, this environment will provide means to organize and manage projects from the students' perspective (e.g., the main screen provides for explicitly listing organizational and learning issues) and the teachers' perspective (e.g., tracking student progress and keeping records of student work). In addition, we envision that the environment will be used for research purposes (e.g., archiving data such as the inter- and intra-group communications and resource sharing that took place during a course for later assessment, collecting data to be used for student/group modeling in order to devise better course- and student-specific on-line scaffolding and coaching methods, etc.). An initial prototype of this environment has been developed with Hypercard on the Macintosh platform, but it has not yet been tested in a classroom. Borrowing from the metaphor of the white board workspace of problem-based learning found in medical schools, this prototype provides an electronic workspace that is split into four regions. It also allows easy access to other tools and resources. Figures 2 and 3 show the workspace and problem screens of this prototype.

Here is a brief scenario to illustrate how we imagine the students will use this environment. Students, who will be working in small groups, enter the environment at the main window shown above, which represents their shared electronic workspace. They are provided with relevant information on the design problem they need to solve via the button "new problem". In this case, it is to design an archery stadium for the Olympics. As students are initially formulating and understanding the problem, they will be encouraged to identify data relevant to the problem from the information they have been provided with, and to articulate this by recording those in the "facts" space. Similarly, as they consider alternative solutions, they will make use of the "ideas" space. The problem-based learning methodology that this environment embodies explicitly prepares students for self-directed learning by requiring them to identify their knowledge deficiencies in the "need to learn" space and the actions they plan to take to remedy those deficiencies in the "action plan" space. Several buttons are found on the bottom of the screen that provide access to different tools that they will need to solve the problem.
"Stage" is a pull-down menu which acts as a gateway to various kinds of software-realized scaffolding tailored to different stages of problem solving.

While the structure of this environment is still evolving, some of its components have already been designed, implemented and individually fielded in classrooms. In the following two sections we elaborate on these implemented components and describe future directions for our research.

4. Implemented Components

**Case Libraries:** Research on case-based reasoning [11] provides guidelines for indexing and making available resources needed while problems are being solved, especially case materials. Case libraries organize cases in ways that make it easy to access their most interesting parts, understand their implications, and recognize the range of problems needing solving and the range of solution methods available. Case studies are structured in terms of overviews, problems, stories, and responses. Each story discusses some problem that arose in designing some artifact, the way that the problem was addressed, and the outcomes that resulted. To make it easy for users to extract from stories their important points, stories are presented with illustrative graphics, and several kinds of contextual information is associated with each story. Students can examine the full artifact that some story is associated with, see a general description of the problem the story addresses, a general description of the kind of solution it provides, and can ask to follow links to other stories that illustrate a similar problem or solution. The stories help students discover which issues they should be considering during design and help them to anticipate the results of carrying out their proposed designs. We have developed a number of case libraries in support of design problem solving.

*Case Library Authoring Tool:* DesignMUSE [5] is a case library authoring tool that has been developed to allow easy construction of case libraries. During the 1995 Winter Quarter it was used to create a library of environmental cases for use in our sustainable technology classes. Thus, while existing case libraries act as intelligent information resources, this authoring tool will allow students to construct their own case libraries to record the design problems that they solve. Both the authoring tool and the case libraries are built on Common Lisp for Macintoshes.

*CaMILE:* Our collaboration software CaMILE [8], based in principle on CSILE [18], integrates information-gathering tools, communication tools, and applications into a collaborative environment. CaMILE provides a discussion environment into which the full range of text, graphics, spreadsheets, video, and so on that reside locally or on the Internet can be incorporated. It is designed to meet two goals. First, it serves as a collaboration and information indexing tool. Discussions are structured and annotated with links to material anywhere across the network. Second, it serves as a design support tool. Discussions about design problems can be annotated with links to actual ongoing designs. The discussion trace can then serve as a design rationale and a case study of a design. It allows students to collaborate in learning and problem solving by providing a facility for structured intergroup and intra-group communications that are archived, and by providing a way to share multimedia documents easily among collaborators. Like CSILE, CaMILE scaffolds collaboration through procedural facilitation. While electronic mail merely allows team members to share ideas, CaMILE helps them to organize their ideas into coherent arguments, relate their ideas to one another, and use resources across the network to support their arguments. CaMILE was built with Hypercard on Macintosh computers.

*Exploratory Simulations:* We have developed a range of exploratory simulations [16] that enable students to learn through simulated experience. Key to these simulations are tight integration with real world problems and activities, and flexible specification of simulation choices to allow for creative and sophisticated simulation problem solution. These simulations have been constructed using the Smalltalk language.

5. Future Directions

**Learning from case libraries:** As students are solving problems, several kinds of resources are needed to help them. Clearly, they need access to documentation of the kind found in books and encyclopedias. But another significant but often overlooked resource is codified prior experience: e.g., cases that describe solutions to similar problems. Our approach to supporting learning from prior experiences is to make on-line case libraries available from within the software environment. Cases help with understanding a problem better, suggesting solutions and parts of solutions, and evaluating proposed solutions, thereby helping a student to know where to focus his/her attention. Our research on case libraries will proceed in two directions. One is generating content: creating the kinds of cases with which to populate these libraries in order to have maximum impact on learning. The other concerns issues of information organization, presentation and navigation. How can cases be organized and presented in ways that make it easy to access their most interesting parts, understand their implications, and recognize the range of problems and the range of solutions available? While the existing case libraries provide one answer to this question (another, for example, is provided by [9]), we are currently revisiting this issue from the perspective of students, who are novice practitioners. From this perspective we believe that additional capabilities such as access to definitions of the terms used by experts, access to explanations of what experts find it
appropriate to focus on, guidance in choosing what to focus on next, and allowing students to extend the libraries (or create new ones) are also required.

**Supporting collaborative problem solving and learning:** Support for group communication and sharing will be provided by facilitating collaborative work through the software environment. CaMILE was used during the past two quarters in a junior-level design foundations course, taught in mechanical engineering (ME 3110; Creative Decisions in Design). We have collected data on the system's usage and its effects, and are in the process of analyzing this data. A World Wide Web version, WebCaMILE, is also under development. We plan to link case libraries and WebCaMILE so that students engaged in a design activity might use WebCaMILE to discuss and exchange case-study materials. Cases provide the kinds of information that a student might point to as justification for some argument presented to others, as a potential alternative to a design decision, or as a rebuttal to someone else's design decision. Tailoring CaMILE's procedural facilitation to reflect more closely the content and nature of the problems students will be solving and investigating new ways of scaffolding collaboration are other topics of ongoing research.

**Software-realized scaffolding:** Of particular importance in making this integrated software environment work for students is providing software-realized scaffolding to support student use of the environment for learning. We have identified several specific areas in which we can provide facilitation.

- **Scaffolding collaborative design and problem-solving:** Our environment will provide scaffolding for design and problem-solving using several techniques:
  - By structuring the kinds of entries which can be made in a group discussion, e.g., new theories or ideas, alternatives, comments, rebuttals, and questions. When a student chooses one of these kinds of entries, an editor opens for their comments and a prompting window opens with suggestions for useful entries to make, e.g., for a rebuttal, suggestions might include "The strengths of this idea are..." and "But the key weakness is...". This scaffolding guides the discussion in useful directions defining the kinds of entries to be made, asking students to choose one before entering an item into the discussion, and suggesting appropriate things to say.
  - By providing agents to actively review student work and suggest better ways to design and solve problems. For example, agents may identify where connections might be made between efforts, where additional resources exist that might aid an effort, and where efforts may be going astray [6].
  - By providing menus of glossaries of relevant vocabulary and their definitions.
  - By providing means of visualization and making explicit the design process.

- **Scaffolding reflection and learning:** We want to support two kinds of reflection in the environment because we believe that reflection can significantly facilitate learning.
  - **Reflection-in-action:** The students' articulations in the discussion, the declaration of item type, and the linking of cases to discussion are all forms of reflection-in-action. These are kinds of reflection which are integral to the design process and which support both the execution of a good design process and the learning about that process. Reflection-in-action helps to make strategies explicit and learnable, develops an expanded repertoire of strategies, and improves student understanding and control of the design process.
  - **Reflection-as-summary:** Student summarization at the end of a design process is an important learning activity for students and an important resource for future groups of students. Our plan is for students to summarize their group design projects such that summaries from one class become cases in the library for the next class. Thus, students summarize not just for their own benefit but to help a future audience.
  - **Scaffolding resource identification and use:** Case libraries support student exploration by providing multiple indices into cases. Students might begin by looking at one case of interest and then explore related cases by a number of different dimensions, or begin by browsing all cases related to a problem. Students can gain perspective on what problems they are facing, what the parameters of the problems are, and how these parameters are explored in the cases in the library from case overviews. We want case libraries to provide support for all these kinds of searching and browsing, but coupled with support that helps in applying the found information to the task at hand (e.g., linking cases that highlight an important alternative solution to the discussion on that alternative). In addition, we envision the use of visualization tools to aid in resource identification and use.

**Integration:** As many of the critical components of the software environment are being implemented and used in classrooms, the most significant task ahead of us is integrating the different pieces into a single environment. This integrated environment supporting the McBAGEL approach has to play several roles: facilitation of design problem solving and its constituent components, facilitation of learning, access to resources, and access to teachers and fellow learners. The software environment has to serve as both an electronic workspace and a learning environment providing help with a variety of intellectual activities as students collaborate on design projects. We see a need for this environment to promote reflection and summarization as well. Software-guided reflection is particularly important in facilitating skill transfer between different problem
domains. The construction of such an environment on Macintosh computers is currently underway.

Assessments: The next step, slated to begin in Fall 1995, is to use and assess both the approach and the concomitant software environment in a series of design courses at Georgia Tech. We will use assessments to determine what kind of learning has occurred and how well students apply what they have learned. The goals of learning involve not merely acquiring a set of static facts to be recalled on a test but rather involve constructing a coherent understanding of a domain that can be flexibly transferred to new situations. The extent to which learning can be used in new situations (i.e., transfer) allows assessment of how flexibly the students have learned the content and are able to apply it to complex problems. Students' learning will be evaluated on mastery, near-transfer, and far-transfer problem-solving. Cognitive research suggests that because problem-based instruction is geared towards complex curricular objectives, assessments need to include open-ended questions in which students explain what approaches they have to a problem and its solution [7]. A variety of methods will be used to collect this data including interviews and paper-and-pencil short answer tests. This allows measurement of the products and processes of the students' learning. Some authentic performance assessments will also be devised. Students' presentations will be assessed to examine how they define the problems and justify their solutions as well as the quality of their solutions. Because transfer is not an all-or-none phenomenon, different types of transfer will be assessed and measures will be developed that assess this. We will use measures of knowledge, skills, planning, and qualitative understanding as students are asked to justify their solutions. This will assess the flexibility of the knowledge that the students construct. For example, because of the emphasis on problem solving, we would expect increased integration of the content they are learning into their problem-solving on transfer problems. Because students are using the collaborative environment and gaining experience and feedback in articulating their plans for problem-solving, we expect improvement in the students' planning skills as well.

6. Conclusions
Collaborative learning environments have the potential for helping students to construct usable knowledge and to learn strategies that prepare them for a lifetime of learning. To afford generative learning, such environments need to contain rich sources of information. In addition, opportunities for student collaboration, articulation and reflection must be provided to help students think deeply about the problems they are working on and to learn to go beyond the given problems. The McBAGEL approach is designed to meet these requirements. Providing computational support to this approach requires the design of a software architecture that integrates multiple tools and information resources with a structured electronic workspace. This paper describes our efforts on developing the theoretical and practical aspects of such an architecture. The focus of our current research is on refining and testing the components further, and on fully implementing the integrated environment. Future research will focus on deploying it in classrooms and conducting assessments of its impact on student learning.

Acknowledgments
Research reported here has been supported by ARPA under contract N00014-91-J-4092 monitored by ONR, ONR under contract N00014-92-J-1234, and by the Woodruff Foundation's support of the EduTech Institute.

References


Authors Addresses
N. Hari Narayanan, Cindy E. Hmelo, Valery Petrushin, Wendy C. Newssetter, Mark Guzdial and Janet L. Kolodner. EduTech Institute, College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA 30332-0280. {narayan, ceh, petr, wendy, guzdial, jlk}@cc.gatech.edu.
Understanding the Creative Mind

Ashwin Ram
Linda Wills
Eric Domeshek
Nancy Nersessian
Janet Kolodner
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

Abstract
We review Margaret Boden’s book *The Creative Mind*, an excellent survey and synthesis of current computational theories of creativity. Boden’s stated goal is to explain how creativity (as a psychological phenomenon) is possible, where an explanation of possibility is taken to be a computational process. Although Boden does not deliver a full-fledged computational explanation and leaves most details of the underlying processes unexplicated, she provides a strong argument that such an explanation is possible.

As part of our critique, we sketch our preferred (case-based) framework for modeling creativity, in which much of mental life depends on the retrieval and manipulation of past experiences. We focus on five major influences on cognition (and thus on creativity): inference, knowledge, task, situation, and strategic control. We also highlight “constructive modeling” which integrates analogical reasoning with visual reasoning and thought experimentation.

Our framework, while broadly compatible with Boden’s, is more specific in its suggestions for integrating multiple types of interacting and interactive processes. We emphasize issues of control and the role of experience. By focusing on how mental activity is directed towards a task in some situation, we ensure that the resulting theory addresses pragmatic issues in thinking and control of thinking.

To appear in *Artificial Intelligence* journal. In addition, a shorter review of Boden’s book by the same authors will appear in the journal *Behavioral and Brain Sciences.*
1 Computational Creativity

Margaret Boden, a master at bringing ideas from artificial intelligence and cognitive science to the masses, has done it again. In *The Creative Mind* (Boden, 1990), she has produced a well-written, well-argued review and synthesis of current computational theories relevant to creativity. This book seems appropriately pitched for students in survey courses and for the intelligent lay public. And if ever there were a topic suitable for bridging the gap between the researchers and the layperson, this surely is it: What is creativity, and how is it possible? Or, in computational terms (the terms that Boden argues ought to be applied): what are the processes of creativity?

Boden’s stated goal is to explain how creativity is possible, where creativity is taken to be a psychological phenomenon, and an explanation of possibility is taken to be a computational process. As computationalists with active interests in creativity, we find this perspective congenial. But while offering many examples of creativity and surveying many approaches to creativity, the book leaves most details of the processes of creativity and their interactions unexplicated. Nevertheless, although Boden does not deliver a full-fledged computational explanation of the phenomenon, she does provide a strong argument that such an explanation is possible.

The early motivational sections of the book enthusiastically play up the notion that creativity, as opposed to “mere novelty,” is somehow paradoxical. The middle section offers broad but shallow coverage of existing computational models with some emphasis on connectionist approaches. The exposition is studded with excellent examples of creativity drawn from the worlds of high culture and epochal science. Major chapters are devoted to “Unromantic Artists” and “Computer Scientists” — that is, to computer programs that have been built to simulate artistic and scientific creation and interpretation (e.g. AARON, BORIS, TAIL-SPIN, ARCS, DENDRAL, BACON, AM, and so on). The final section dwells on a grab-bag of ancillary issues such as the relationship of randomness to creativity, the degree to which creativity is reasonably conceived as a special gift limited to the chosen few, and the nature of computational theories and explanations of phenomena such as creativity (including a round in Searle’s Chinese Room).

Our interest is primarily in the examples, implementations, and theories that comprise the middle section of the book, but it is important to spend a bit of time understanding the problem as Boden has laid it out. As noted, Boden’s goal is a computational account of the psychological phenomenon of creativity. Her achievement is to make the possibility of such a theory seem more probable (or, perhaps, at least conceivable). Both goal and achievement, however, must be contrasted with other possible ends. Boden designates the object of her study as “P-creativity” (or *psychological creativity*), distinguishing it from at least two other related concepts: “mere novelty” and “H-creativity” (or *historical creativity*).

P-creativity is a cognitive notion. By asking how some individual came up with an idea that seems beyond what they ought to be able to think, one concerns oneself with thought processes, and can deploy all the tools of computational modeling to understand these processes. In contrast, H-creativity refers to judgments that are made by a culture about the novelty and worth of ideas. Boden downplays the value of this standard, arguing convincingly that H-creativity is overly restrictive, and that P-creativity is the more significant in that H-creativity typically results from it. Boden chooses many H-creative ideas as
glamorous examples, but the assumption remains that most instances of H-creativity must in the end be explained in terms of some individual's P-creative act. We agree that the important scientific question is how P-creativity could happen and that the right kind of answer to this question is a computational one. After all, no one has much of a handle on a computational model of culture. The key distinction between P-creativity and H-creativity is Boden's position that creativity is an attribute of mental processes rather than mental products. Although there is consensus that historically significant innovations are creative, Boden holds that what is creative when thought by one individual may not be so when thought by another. As computationalists, we like this emphasis on process over product in defining creativity.

Furthermore, we believe that a creative outcome is not the outcome of extraordinary mental processes, but of mechanisms that are on a continuum with those used in ordinary thinking. In our view (and Boden's), extraordinary outcomes arise from the application of ordinary mechanisms, enhanced and applied with conscious (strategic) control. For example, later in this review we describe Maxwell’s use of analogy in deriving the electromagnetic field equations. In doing so, Maxwell constructed a hybrid analogical source model for electromagnetism that draws physical and mathematical constraints from two mechanical source domains: continuum mechanics and machine mechanics. It is not ordinary to construct a hybrid hypothetical analogy as Maxwell did, but analogy is an ordinary mechanism. To understand creativity, we need to understand what is different about the employment of ordinary mechanisms in creative problem solving. The focus on the outcome, for example, as in historical creativity, may provide criteria for what counts as a creative idea, but not an understanding of what is a creative reasoning process. (We will return to this point later.)

Boden's distinction between psychological and historical creativity is important (in fact, indispensable) in establishing the book’s focus. Opposing P-creativity to mere novelty is also important. It serves to rule out easy, boring cases of new ideas that are not interestingly new. How Boden makes this distinction, however, strikes us as somewhat problematic. Boden argues that true creativity (as opposed to mere novelty) occurs when a person thinks a thought that is outside the space of thoughts that are even conceivable to that person—outside, as it were, their knowledge level (Newell, 1982). To clarify this idea, she invokes representations, rules, and search spaces, noting that fixing these constructs limits what can be generated by the thought processes of the reasoner. Creativity, then, requires the modification of these structures in order to expand their generative capacity.

Notice, however, that these clarifications have the effect of building aspects of a particular computational account of mental life into the definition of creativity. The effect is to limit the range of computational explanations up for consideration to those that are expressible within the particular computational paradigm chosen to model the mind. We believe the choice of constructs playing a role in mentation (and thus up for modification) are subject to debate; as will be elaborated below, we would invoke constructs such as cases, indexing structures, adaptation rules, and control strategies.

The balance of this review is organized as follows. In the next section we offer an initial critique of Boden’s approach to characterizing creativity, and raise a set of questions we believe must ultimately be addressed (though we certainly do not claim to be able to answer all of them). In Section 3, we lay out our preferred framework for thinking about and modeling creativity—a framework in which much of mental life depends on the retrieval
and manipulation of past experiences. Within this case-based reasoning framework, we focus on five major influences on cognition (and thus on the potential for creativity); each of these five influences is illustrated using examples of mechanical design, the first of three different domains we have studied, and is related to some of Boden's observations. Section 4 takes up some of the issues raised in our early critique of Boden's model, using examples of everyday creative interpretation (our second domain) to argue against the notion of special creative processes. Section 5 focuses on "constructive modeling," which integrates analogical reasoning with visual reasoning and thought experimentation. The value of this process and how it fits into our framework is illustrated by an example of historically (and psychologically) important scientific creativity (our third and final domain). Section 6 concludes this review by summarizing our approach to modeling creativity and relating it to Boden's position.

2 Characterizing the "Thinkable"

Although we disagree with Boden's choice of constructs, one needs some characterization of the space of thoughts that are ordinarily thinkable by the computational model, and the set of modifications to the thought-generating elements in the model that modify this space in an interesting manner. Ideally, what counts as an interesting modification should be specified in a manner independent of the particular computational modeling paradigm. although the modification mechanisms themselves can, of course, only be specified in the chosen formalism. In other words, the issue is: whatever the constructs involved in mentation, be they cases, rules, or search spaces, what counts as the "ordinarily thinkable." and what counts as a "creative" (as opposed to mundane) modification of the space of ordinarily thinkable thoughts?

We agree with Boden in that she refrains from defining the thinkable in terms of what is derivable through deduction from the reasoner's knowledge (as, for example, is often done in formalizations of "knowledge levels" (Dietterich, 1986; Newell, 1982)). Instead, the search space includes everything derivable from all the available reasoning operators (which could, and usually do, go beyond deduction). However, this leads to the paradox that, in some sense, every thought must be part of the set of thoughts that could be generated through available reasoning operators; if one comes to think a thought, it must have been thinkable. Boden's answer to this is that some operators carry out conceptual change (Carey, 1985; Nersessian, 1992; Ram, 1993; Thagard, 1992) and thus fundamentally modify the search space.

This account falls short in two ways: first, conceptual change is as elusive a notion as creativity itself (Nersessian, 1992), and second, it is not obvious why the search space generated by application of conceptual change operators is not considered part of the thinkable. An independent (and operationalized) characterization of what makes these conceptual change operators different from all the other more ordinary inferential operations is needed.

In particular, consider Boden's formulation of thought as a search over a given search space defined by a set of constraints, operators, and representations. Boden implies that creative search involves changing or extending the constraints, operators, or representation, using an additional set of operators (with associated constraints and representations) whose job it is to modify the first set. Thus, ordinary thought is a search over an ordinary (albeit
non-deductive) search space, whereas creative thought is a *meta-search* using a separate set of operators. While such an account, in principle, is perfectly acceptable, it is unclear what theoretical principles would license the placement of a given operator (or piece of knowledge) into one or the other of the search or meta-search categories. As we will elaborate below, we do not believe there are special meta-search operators that are different from ordinary inferential mechanisms.

Furthermore, we are skeptical that those individuals noted for producing many interesting ideas undergo radical conceptual change in order to produce each idea. Although this may be true of many historically significant ideas, we would prefer a model of *long-term conceptual development* in which the individual evolves a search space, that, when explored by *normal* thought processes, still includes many thoughts that would be considered creative.

These objections notwithstanding, we are fully sympathetic with Boden's goal of explaining creativity by appeal to computational processes. We were, therefore, most interested in the particular set of processes suggested: heuristic search (as in BACON), multiple levels of representation (as in BORIS), fuzzy matching (using an unspecified connectionist implementation), and most notable, conceptual change (unimplemented).

We agree with the idea of creativity emerging through multiple interacting processes, but we think that Boden's account leaves open several questions. First, the discussion of the mechanisms, though suggestive, is more descriptive than computational. Second, it is unclear what the overall process model is: How do all these mechanisms fit together? How do they interact? Do they operate on the same representations? If not, how do they communicate, and what do they communicate about? A third set of issues relates to Boden's suggestion that these processes are not unique to specially endowed individuals. It is never quite clear whether these processes are unique to creative thought, or, if not, what distinguishes those thoughts that are creative from those that are not, within a single individual.

### 3 Five Aspects of Thought

Parallel to and independent of Boden's analysis, we have been studying creative reasoning in several different domains, with a similar goal of producing computational process models of creativity. Much of what we have found concurs with Boden's observations and proposals, but we are seeking more specifics and more coherence in our models. We believe that in order to analyze creative reasoning, one needs a theoretical computational framework in which to model thinking. To this end, we propose using a computational approach rooted in case-based reasoning (Kolodner, 1993). This paradigm is fundamentally concerned with memory issues, such as remindings from partial matches at varying levels of representation and the formation of analogical maps between seemingly disparate situations—exactly the kinds of phenomena that researchers up to, and including, Boden have highlighted as central to creativity.

Accordingly, we see creative thought, like all thought, as involving processes of problem interpretation and problem reformulation, case and model retrieval, elaboration and adaptation, and ultimately, evaluation. Interpretation and reformulation are part of *situation assessment*—the process of redescribing a problem in the vocabulary of a memory's indexing scheme. Elaboration and adaptation include standard analogical processes as well as the
more general process of constructive modeling, discussed at length in Section 5. Evaluation includes outcome determination, be it by simulation or by case-based prediction. All of these processes follow from our enriched case-based reasoning model (Kolodner, 1994), and fit together into a coherent whole within that framework. Research in case-based reasoning has provided extensive knowledge of how to analyze and reformulate problems, how to reuse solutions to old problems in new situations, how to build and search libraries of experience, how to merge and adapt experiences, and how to evaluate candidate solutions.

Our examples of creativity are drawn from three disparate domains: We are studying creativity in the everyday activities of average people by studying the design of mechanical devices (Kolodner & Wills, 1993a; Wills & Kolodner, 1994a, 1994b) and by looking at the processes involved in reading and understanding science fiction stories (Moorman & Ram, 1994a, 1994b; Ram, 1993). At the same time, we are examining and analyzing what led to the significant scientific discoveries of Maxwell and Faraday (Nersessian, 1984, 1992, 1993). Examples drawn from these studies, as well as Boden’s own examples, will be used to illustrate our points.

Our research suggests that creativity is not a process in itself that can be turned on or off; rather, it arises from the confluence and complex interaction of inferences using multiple kinds of knowledge in the context of a task or problem and in the context of a specific situation. Much of what we think of as creativity arises from interesting strategic control of these inferences and their integration in the context of a task and situation. These five aspects—inferences, knowledge, task, situation and control—are not special or unique to creativity but are part of normal everyday thinking. They determine the thinkable, the thoughts that the reasoner might normally have when addressing a problem or performing a task.

To give a taste of what we mean by each of these five aspects, the next five sections give examples of each aspect in the context of design. Design is a pervasive form of thinking which most people do every day, not just in specific engineering contexts. All five aspects of thought are involved in design reasoning along the entire continuum from routine to creative design. The goal of this section is to give examples of the five aspects that determine the thinkable. The next section discusses what it means to go beyond the thinkable with respect to these five aspects.

### 3.1 Inferential Mechanisms

We have performed an exploratory study in which we observed a four-person team engaged in a seven-week undergraduate mechanical engineering (ME) design project (Kolodner & Wills, 1993a; Wills & Kolodner, 1994a). The task was to design and build a device to quickly and safely transport several eggs from one location to another. In this study, we observed that designers move fluidly between a variety of inferential methods. Typical ones include problem understanding, decomposition, elaboration, and redescription, as well as remembering, adapting, and merging design artifacts previously seen.

For example, while trying to think of ways of launching a heavy transport device, carrying several eggs from a pool of water, our ME designers recalled the behavior of a submarine submerging and launching a missile. This helped them to visualize the desired behavior of the device being designed and to elaborate the problem specification. While visualizing and
acting out the missile launch, the students noticed that submarines launch missiles one at a time. This led to a redescription of the problem from launching a group of eggs in a single launch to launching each egg individually in multiple launches. The students went on to merge this idea with other ideas they had earlier, such as enclosing each egg in a tennis ball for protection (an adaptation of an earlier idea to enclose several eggs in a NERF football).

Such inferences are driven and guided by the evaluation of proposed design ideas through critical analysis, as well as by experimentation and mental simulation. The generative mechanisms, guided by critiques, respond to opportunities to create new alternatives by merging or adapting proposed ideas. The design specification is incrementally updated as ideas are tested and flaws or desirable features become apparent.

The types of inferential methods we observed (e.g., problem elaboration and redescriptions, solution remembering, adapting, and merging) were applied throughout the design process to produce routine (thinkable) as well as innovative ideas. They were applied in a flexible and highly opportunistic manner, with their application heavily influenced by the other four aspects of thought. Computational models of several inferential mechanisms exist, which exemplify the inferential aspect of thought. These include:

- reinterpretation of an idea in terms of a different but familiar idea (e.g., Jones (1992) shows how this can lead to useful problem reformulations which facilitate the operationalization of abstract advice (in the form of proverbs) during planning situations).
- visualization, mental simulation, and thought experimentation, which we have seen to be useful in evaluating and elaborating ideas, and in reformulating problems in design (Kolodner & Wills, 1993b) and scientific reasoning (Nersessian, 1992, 1993).
- constraint relaxation and substitution, which is useful in problem reformulation and elaboration (e.g., Moorman & Ram (1994a) show how new concepts can be formed or understood, while reading science fiction stories, by systematically tweaking constraints on known, familiar objects),
- relaxing constraints during memory search, which facilitates problem reformulation and retrieval (e.g., Turner (1994) calls this imaginative retrieval and shows how it can be used to retrieve ideas for writing short stories),
- relevance assessment, which is useful, for example, in retrieval and evaluation (Ram & Leake, 1991), and
- explanation of anomalies, which is also useful in retrieval and evaluation (e.g., (Ram. 1994; Schank, 1986)).

3.2 Knowledge Sources

Our second aspect of thought is knowledge. Designers draw on a variety of knowledge sources, particularly previous design experiences, accumulated from personally designing artifacts, studying case studies of designs in school, and observing artifacts designed by others. Designers typically work within a “design culture” (Navinchandra, 1992) of common engineering practices, design styles, techniques, and technologies. Innovation often arises when ideas from one culture are applied in another. In our ME design study, one designer
drew much inspiration from automotive engineering, a design culture in which he is intensely interested. Many of his ideas came from recalling devices and concepts from the car domain, such as shock absorbers, unit-body versus single-frame construction, and air-bags.

A crucial part of what makes this transfer possible involves understanding, elaborating, and redefining the given problem specification to make connections to domains with which they are familiar. Designers often build on their knowledge of previous, similar problems (and their solutions) to derive new constraints and priority structures that improve or go beyond those stated in the original problem description. For example, our ME designers redefined their launch problem, based on recalling how submarines launch missiles. They derived evaluative issues and new criteria and constraints, based on their experiences with devices such as cars, toys and sports equipment, as well as designs for previous high-school egg-drop projects.

Many of the aspects of constraint exploration we observed in our designers can be experienced by Boden's reader when, in Chapter 4, she encourages the reader to play a game of necklace building within a set of rules. As Boden points out, the construction and exploration of conceptual spaces is often facilitated by drawing analogies to familiar concepts so that knowledge and reasoning techniques can be transferred to the current problem. As we will show later, the same sorts of redescription and construction of conceptual structures occur in the other two areas we have studied—science fiction reading (in which new concepts must be invented to understand the stories) and scientific discovery (in which new hybrid models are designed by merging pieces of knowledge from multiple source domains). We call this process constructive modeling (Clement, 1989; Moorman & Ram, 1994b; Nersessian, 1992, 1993, in press; Nersessian & Greeno, in process). Other existing mechanisms for accessing and manipulating knowledge sources include redescription and abstraction, such as reinterpretation of data at a higher level (for example, symbolic interpretation of numerical data (Ram, 1993; Kuipers & Byun, 1991)), and cross-contextual analogy (e.g., (Ram, 1993; Schank, 1982)).

Transferring knowledge from one design culture (or domain, in general) to another is not necessarily P-creative. However, identifying a domain as relevant, figuring out which pieces of knowledge or which strategies can be transferred to a new problem, and how to adapt and combine them to solve the new problem can be a creative process. These are important questions of focus which Boden does not address, but which are central to understanding what guides exploration within a generative system. (Boden is concerned more with how creativity is possible than with what guidance can make it more probable.) We believe many of the answers to these focus-related questions come from the task at hand and the situational context.

### 3.3 Tasks

A third aspect influencing what is thinkable is the task. Design is a complex task, involving several subtasks, such as brainstorming, critiquing, gathering information about and elaborating ideas, and finding, constructing, and integrating design pieces. Which aspects of a remembered design experience or a proposed design alternative the designer focuses on depend on what is relevant to the task at hand. This can greatly influence the strategic control of the design process, as well as which new constraints or criteria are added to the
design specification and which elaborations or adaptations of ideas are suggested.

For example, there are numerous facts associated with submarines, but our designers were drawn to the fact that they launch missiles one at a time, as opposed to, for example, facts about how missiles are aimed at their target or about the cramped, claustrophobic interior. They were viewing the submarine missile launch from the perspective of trying to borrow its solution to the problem of initiating a powerful launch from water; thus, what was relevant was the detail that multiple, relatively small missiles are launched one at a time. This focus on individual launches helped suggest a new way of looking at the problem (Kolodner & Wills, 1993b).

3.4 Situation

Situation is our fourth aspect of thought. Design does not typically occur in a vacuum. Rather, designers usually try to experiment with their design (e.g., a mock-up, simulation, prototype, or partial construction) in a real-world situation (e.g., the typical operating environment, a potential maintenance situation, a worst-case scenario). This provides concrete feedback that can refine the problem specification to require any positive features noticed and to prohibit any flaws that were detected. At the same time, the evolving specification can be used to reinterpret entities in the environment and realize their relevance to the problem at hand.

Designers operate in a rich context of ideas, which are not only recalled and adapted from previous experiences, but also recognized in the current external environment. (That is, the environment can be a source of inspiration, in addition to knowledge and experiences recalled.) The continual elaboration and reformulation of the problem and desired solution primes the designer to recognize good ideas when they are stumbled upon. Problem redescription often enables the designer to overcome functional fixedness and notice new, alternative functions and uses for common design pieces. This leads to insights into new ways of solving pending problems (thus facilitating serendipity).

For example, at one point in the ME design project, the students were considering using a spring launch device, but had the problem that the springs bent when compressed. After generating, simulating, and critiquing a few proposals, they augmented their specification to require that each spring be enclosed in a collapsible tube. However, they could not immediately think of anything that could serve as a collapsible tube, so they temporarily gave up on designing the launch mechanism. Later, as they were looking for protective egg cushioning material, they came across toilet paper holders and immediately recognized them as the collapsible tubes they needed to keep springs straight (Wills & Kolodner, 1994b). By playing with the springs, noticing problems, and suggesting fixes, the designers formed a specific, concrete description of what they needed. This description was used to reinterpret the paper holder when it was seen and to recognize its additional function of preventing springs from bending upon compression.

Being situated facilitated the designers’ discovery by bringing to their attention objects that could solve their problem without requiring the objects to be recalled as relevant solutions. Playing with the springs in a concrete situation also provided feedback to help the designers elaborate and refine their description of what they needed. The designers became immersed in the problem – redescribing it and viewing it from multiple perspectives.
considering, comparing, and critiquing several options – so that when a relevant solution was spotted, the way it fit into the problem was immediately discerned.

The importance of becoming immersed in the problem situation is implicitly acknowledged by Boden when she interrupts Chapter 4 to encourage the reader to temporarily stop reading and to play the necklace-building game. She suggests that the reader practice building necklaces (with pencil and paper), play around with the rules, record any interesting things that are noticed, etc. Although Boden does not analyze why this is so important, constructing specific necklace-building situations does provide feedback that can help the reader understand the problem constraints, their implications, and ways of modifying them.

3.5 Strategic Control

Finally, the fifth aspect of thought is the strategic control of inferences. Designers must make many decisions over the course of a design: which idea to elaborate or adapt next, which constraint to relax, how to set priorities. They also move between various tasks, subproblems, and design processes in a flexible and highly opportunistic manner.

We observed a variety of strategic control heuristics used by our ME designers. Some were opportunistic. An example is letting extremes distract. When an alternative was proposed that satisfied some desired criteria extremely well compared to the other alternatives, our designers directed their efforts toward elaborating that alternative (Wills & Kolodner, 1994b). They optimistically suspended criticism or discounted the importance of criteria or constraints that were not satisfied as well. Suspending criticism during brainstorming is a common strategic ideation technique which involves taking a cognitive risk. A similar mechanism is seen in creative interpretation, in which the reader must suspend disbelief in unfamiliar aspects of a story in order to understand it (see below). Sometimes, as constraints are relaxed or placed at a lower priority, an opportunity to reformulate the problem is revealed (Kolodner & Wills, 1993b). Noticing invariants (Kaplan & Simon, 1990), as well as anomalies, can also aid in understanding a problem and reveal ways of redescribing it.

Some strategic control heuristics are more deliberate, based on reflection. For example, one heuristic our designers used was to try quick, easy adaptations of a proposed solution first before stepping back and reformulating the problem or relaxing constraints (Wills & Kolodner, 1993a, 1994a). Other deliberate heuristics include making non-standard substitutions (Kolodner, 1994; Kolodner & Penberthy, 1990), applying adaptation strategies in circumstances other than the ones they were meant for (Kolodner, 1994; Navinchandra, 1992), merging pieces of separate solutions with each other in nonobvious ways (Kolodner, 1994; Kolodner & Penberthy, 1990), and goal-directed inferential control (Nersessian, in press; Ram, 1991; Ram & Hunter, 1992).

Often, creativity arises when a set of “normal” strategies are applied to a situation in which a run-of-the-mill solution is not immediately forthcoming and the control heuristics allow the reasoner to devote more resources to the problem, looking further and further afield for possible knowledge and strategies until something results in a creative solution. Examples include a problem reformulation that takes several steps; an analogy to a far-off case or model; an analogy from a hybrid analog constructed incrementally from more than one source; a strategy imported from a different problem-solving culture; an unexpected and novel opportunity afforded to the reasoner by virtue of an unusual task context. Many
of these could happen during “ordinary thought,” but most thought does not allow enough leeway to look that far or to play with ideas for that long or it does not occur in a context that affords such an opportunity.

4 Beyond the Thinkable

Based on this view of creative thought, we offer a very pragmatic definition of the *normal* search space. It is not the deductive (or other) closure of everything that is known – an inherently uncomputable concept. Rather it is the *space of the thoughts one would usually explore in a pragmatic context*. There may be cases where important possibilities are outside the space of theoretically conceivable thoughts. (Perhaps rings of carbon atoms could never arise within the chemical theory prevailing at the time Kekule tackled benzene.) But, in other cases, thoughts that are within the theoretical space are nevertheless pragmatically inconceivable (e.g., the discoveries made by Swanson’s (1990) program which are nevertheless H-creative). In creative individuals, even the usual search space may be interestingly different or expanded so as to provide the basis for creative thought using the very same mechanisms that on other occasions would produce more mundane thoughts.

Consider, for example, the problem of reading a science fiction story. Although creativity is usually thought of in the context of problem-solving or inventive tasks, we believe that creativity is an essential and ubiquitous component of other kinds of reasoning tasks as well, including explanatory and comprehension tasks. In point of fact, all these tasks involve understanding. Reading science fiction stories requires what we call *creative understanding*, in which the reader must learn enough about an alien world in a short text in order to accept it as the background for the story and simultaneously must understand the story itself. Creative understanding requires the extrapolation, modification, or extension of existing concepts and theories to invent new ones (Moorman & Ram, 1994a, 1994b; Ram, 1993). The extrapolation is constrained by the content of the story, by the system’s existing concepts and theories, and by the requirements of the reading and understanding task.

As an example, consider the following short story, *Men Are Different* by Alan Bloch (1963).

I’m an archaeologist, and Men are my business. Just the same, I wonder if we’ll ever find out about Men – I mean really find out what made Man different from us Robots – by digging around on the dead planets. You see, I lived with a Man once, and I know it isn’t as simple as they told us back in school.

We have a few records, of course, and Robots like me are filling in some of the gaps, but I think now that we aren’t really getting anywhere. We know, or at least the historians say we know, that Men came from a planet called Earth. We know, too, that they rode out bravely from star to star; and wherever they stopped, they left colonies – Men, Robots, and sometimes both – against their return. But they never came back.

Those were the shining days of the world. But are we so old now? Men had a bright flame – the old word is “divine,” I think – that flung them far across the night skies, and we have lost the strands of the web they wove.
Our scientists tell us that Men were very much like us – and the skeleton of a Man is, to be sure, almost the same as the skeleton of a Robot, except that it’s made of some calcium compound instead of titanium. Just the same, there are other differences.

It was on my last field trip, to one of the inner planets, that I met the Man. He must have been the last Man in this system, and he’d forgotten how to talk – he’d been alone so long. I planned to bring him back with me. Something happened to him, though.

One day, for no reason at all, he complained of the heat. I checked his temperature and decided that his thermostat circuits were shot. I had a kit of field spares with me, and he was obviously out of order, so I went to work. I pushed the needle into his neck to operate the cut-off switch, and he stopped moving, just like a Robot. But when I opened him up he wasn’t the same inside. And when I put him back together I couldn’t get him running again. Then he sort of weathered away – and by the time I was ready to come home, about a year later, there was nothing left of him but bones. Yes, Men are indeed different.

In order to understand this story, the reader must infer that the narrator is a robot, that robots are the dominant lifeform in the future, that humans have practically died out, that robots are capable of making factual errors such as the ones that the narrator made, and so on. The reader must construct an appropriate model of this world, and interpret the story with respect to this model even as the model evolves. The reader must also be willing to suspend disbelief (Corrigan, 1979) to understand concepts which do not fit into a standard world view. This is another example of a strategic control mechanism that requires a willingness to take a cognitive risk.

In *Men Are Different*, robots, which in the real world are physical objects used as tools in manufacturing, are conceptualized as independent volitional agents. The new concepts are constructed by merging and extending the existing concepts representing human agents and robotic artifacts, resulting in a novel view of the situation at hand (Moorman & Ram, 1994a). The reader must adopt this view to build an appropriate story model. Interestingly, the irony in this story derives from the fact that the robot in the story performs what one might view as the reverse inference: conceptualizing the man as a physical object to be repaired in a manner that one might use to repair a physical robotic device (Moorman & Ram, 1994b).

It would, of course, be unreasonable to assume a special purpose “meta-search space” generator for science fiction story understanding. The creative understanding processes required to read *Men Are Different* are not unique to science fiction stories; understanding any fictional story requires similar kinds of processing. The same is true of nonfictional stories as well as unfamiliar real-world scenarios, although the types and degree of conceptual modifications required may be different.

Thus, reading a science fiction story is presumably accomplished within the same type of search space and using the same set of reading and comprehension operators as reading a mundane narrative. The example illustrates that these ordinary operators and processes can take the reasoner out of the space that would usually be explored. In fact, situations like this show just how fluid the movement is from the usual to the unusual.
The question, of course, is how the search space comes to be expanded to facilitate creative thought using ordinary mechanisms. If normal traversal of a search space depends on knowledge, inferential methods, and control methods, then interesting paths may result from modifying any of these three components. Most obviously, transformations of basic knowledge (e.g., conceptual change) can yield new results. But application of new inferential methods can also produce novelty; for example, adopting a heuristic from a different task context, such as an architect adopting the engineer’s heuristic of “incorporate the obstacle.” Finally, differences in control methods will produce differences in results; consider methodological differences between scientists, such as the willingness to take cognitive risks, the willingness to explore a “silly” idea, the ability to evaluate and prune unlikely candidates. For example, we would rate AM+Lenat as a creative combination even though AM by itself was not. Analysis of the task and situation influences the knowledge, inferential methods, and control strategies that are available.

5 Constructive Modeling

Reading and understanding *Men are Different* requires the invention of a system of concepts and theories that represent a sentient, humanoid robot, through the extension of one’s prior understanding of multiple concepts, such as volitional agents, men, and industrial robots (Moorman & Ram. 1994a). In creative design, too, new conceptual structures are formed from multiple sources. Problem descriptions are incrementally elaborated and reformulated, typically by analogy to pieces of several similar problems. New design ideas are generated by combining several ideas from experiences with existing devices. The behavior of a proposed design is predicted, simulated, and visualized based on multiple pieces of knowledge of how related devices or design pieces work.

These are everyday instances of the constructive modeling process we have found to be central in significant scientific discoveries throughout the history of the sciences. For example, it figures centrally in the development of the field representation of electromagnetic forces by Michael Faraday and James Clerk Maxwell. Here we will illustrate our points by looking briefly at Maxwell’s derivation of the electromagnetic field equations (Maxwell 1890). The Maxwell case reinforces Boden’s contention that even in instances of H-creativity, explaining the episode demands an analysis of P-creativity.

This case shows constructive modeling to be a dynamic process involving analogical and visual modeling as well as thought experimentation (mental simulation) to create sources where no direct analogy exists (Clement, 1989; Nersessian 1992, 1993, in press; Nersessian & Greeno, in process). What distinguishes this process from the computational models of analogical reasoning Boden discusses is that they employ cases where the analogical base is ready to hand. Further, although Boden does note the importance of visual representation in some instances of analogy, neither she nor the computational models she discusses attempt to integrate it into their accounts. Indeed, we believe the constructive modeling processes identified in the Maxwell case show the need for an integrated account of analogy, visual representation, and mental modeling for understanding creative thinking.

Finally, this case points to something missing entirely from Boden’s analysis. The social context is crucial to understanding a creative episode in science - and we presume in more ordinary cases, too. Maxwell’s location in Cambridge led to his training as a mathematical
physicist. This determined the nature of the theoretical, experimental, and mathematical knowledge and the methodological practices with which he formulated the problem and approached its solution. The work of Faraday and William Thomson (later, Lord Kelvin) contributed to these as well. Continental physicists working on electromagnetism at the same time employed quite different practices and drew from fundamentally different mathematical and physical representational structures. These kinds of social factors can be figured into the account without our being required to produce a computational model of culture.

Maxwell’s constructive modeling process provides a good example of an instance in which all five of the aspects of creative thinking we have been discussing are employed. He used multiple knowledge domains and informational formats, in the context of solving a complex problem within specific cognitive and social situation. Maxwell exercised strategic control continually to evaluate the models and the inferences he drew from them, and to integrate the solutions to the sub-problems into a consistent mathematical representation. The modeling process involved adjusting multiple constraints drawn from

- the physics of elastic fluids,
- experimental data on electricity and magnetism,
- Faraday’s hypotheses about the lines of force that form when iron filings are sprinkled around magnets and charged matter (Faraday, 1835-55),
- Faraday’s visual lines of force model (shown in Figures 1a and 1b), accounting for continuous transmission and interconversion of forces (Maxwell, 1890, vol. 1, pp. 155-229),
- Faraday’s interlocking curves model (shown in Figures 2a and 2b), representing the dynamical balance between electricity and magnetism (Maxwell, 1890, p. 194n), and
- William Thomson’s hypothesis of rotational motion of magnetism and his analogies and mathematical equations (Lamor, 1937).

Maxwell’s goal (Maxwell, 1890, vol. 1, pp. 451-513) was to provide a unified representation of the continuous transmission of electric and magnetic forces that he hoped would encompass optical phenomena as well. The full model is an imaginary hybrid construction that integrates physical and mathematical constraints from two analogical source domains—continuum mechanics (fluids, elastic media, etc.) and machine mechanics—with constructs from magnetism and electricity. Unlike the cases customarily considered in the literature on analogy, where an existing problem solution in the source domain is transferred to the target domain, in this case, the source and target domains interact to create and modify a series of constructed models that become the objects with which Maxwell reasoned (Nersessian, in press; Nersessian & Greeno, in process). Further, reasoning with the models demands that they provide simulations and thus be animated in a manner similar to thought experiments (Nersessian 1993). In the text itself, Maxwell provided an extensive set of instructions for how the reader should visualize and animate the models.

Maxwell’s model construction proceeded as follows. Maxwell first constructed a primitive model (Figure 3a) consistent with the constraints discussed above: a fluid medium composed of elastic vortices and under stress. With this form of the model he was able
to provide a mathematical representation for several magnetic phenomena. Analyzing the relationships between current and magnetism required alteration of the model. We can see in Figure 3a that all the vortices are rotating in the same direction, which means that since they touch, friction is produced and they will eventually stop. Mechanical consistency, thus, requires the introduction of "idle wheels" (as in machine gears) surrounding the vortices, and Maxwell argued that their translational motion could be used to represent electricity. Figure 3b shows a cross section of the hybrid model. For the purposes of calculation, Maxwell now had to make the elastic vortices into rigid pseudospheres. We can see how the imaginary system provides a mechanical interpretation for electromagnetism: motion of the particles creates motion of the vortices and vice versa. In this model, as was known experimentally, electric current produces magnetic effects and changes in magnetic effects produce current. Using the model, he derived mathematical equations to represent these relationships.

It then took Maxwell nine months to figure out how to represent the final – and most critical – piece of the problem: electrostatic actions. He found that if he made the vortices elastic and identified electrostatic polarization with elastic displacement, he could calculate the wave of distortion produced by polarization. That is, adding elasticity to the model enabled him to show that electromagnetic actions are propagated with a time delay, i.e., they are field actions and not Newtonian actions at a distance. At this point, we have a fully mathematized representation of the electromagnetic field. There are significant sign "errors" in this part of Maxwell's analysis, but Nersessian (1984, in press) has argued that all but one (a minor substitution error) can be seen not to be errors when we view him as reasoning via the constructed model.

This case study illustrates that it was through a process of embodying physical and mathematical constraints in a series of constructed models and reasoning about and with these that Maxwell generated the field equations for electromagnetism – an historically and individually creative process.

6 Summary and Conclusions

Inference and the control of inference, knowledge representation and representational change: these are the main interrelated pieces of the creativity puzzle. Each relies heavily on episodic and semantic memory. Together, they fit into a model of reasoning that is recognizable as (but looser than) case-based reasoning. A creative individual is one in whom these factors combine to form a search space – a repertoire of thoughts – that is different from the usual and contains many creative ideas waiting to be constructed. Of course, the search space can only be explored in the context of a task or problem and a specific situation; thus, the repertoire is defined pragmatically, and serendipity (as Boden points out) plays an important role.

In a specific individual, more creative thoughts will likely result when these pieces come together in a novel way to yield an unexplored and unexpected path through the search space. Creativity, as Boden points out, is not an all-or-none phenomenon. Every new thought is creative to some extent. Every new thought results from those same processes that, on occasion, produce results we value as creative. The more the search space is varied in a given context (through representational change, novel inferential methods, or strategic
control heuristics), the more creative the resulting thoughts are likely to be. Over time, an individual may become more expert as he or she acquires (or reformulates) knowledge, reasoning strategies, and methodologies that change the search space or how it is explored.

The framework we have sketched here is broadly compatible with Boden's, but is more specific in its suggestions for integrating multiple types of interacting and interactive processes in a task context. In accounting for creativity, we emphasize issues of control and the role of experience (or cases). By focusing on how mental activity is directed towards a task in some situation, we ensure that the resulting theory addresses pragmatic issues in thinking and control of thinking. As Boden would require, our approach is computational. We believe, in fact, that the greatest contribution of The Creative Mind is the clear case it presents for the legitimacy of computational theories of creativity. Boden leads the reader to an understanding of that goal, and, having framed the question, suggests how research might proceed towards a meaningful answer.

7 Acknowledgments

This paper is based in part on research by Kenneth Moorman, who also provided us with helpful suggestions for improving this paper. We are grateful to our editors Mark Stefik and Stephen Smoliar for their many insights and helpful suggestions for improving this review. This work has been supported in part by the Advanced Research Projects Agency, monitored by ONR under contract N00014-91-J-4092, by NSF Grant No. IRI-8921256 and ONR Grant No. N00014-92-J-1234, by NSF Scholars Awards DIR8821442 and DIR9111779, and by the Georgia Institute of Technology. All views expressed are those of the authors.

8 Bibliography


17


Nersessian, N. & Greeno, J. (in process). Dynamic mental modeling in scientific reasoning, manuscript.


