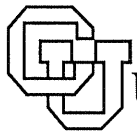
SUPPORTING COLLABORATIVE DESIGN BY EMBEDDING COMMUNICATION AND HISTORY IN DESIGN ARTIFACTS

Brent Neal Reeves

CU-CS-694-93 December 1993



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

SUPPORTING COLLABORATIVE DESIGN BY EMBEDDING COMMUNICATION AND HISTORY IN DESIGN ARTIFACTS

CU-CS-694-93 December 1993

Brent Neal Reeves

Department of Computer Science
University of Colorado at Boulder
Campus Box 430
Boulder, Colorado 80309-0430 USA

Supporting Collaborative Design by Embedding Communication and History in Design Artifacts

by

Brent Neal Reeves

B.B.A., Abilene Christian University, 1980

M.A., Abilene Christian University, 1982

M.S., University of Colorado, Boulder, 1991

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

1993

Reeves, Brent Neal (Ph.D., Computer Science)

Supporting Collaborative Design by Embedding Communication and History in Design Artifacts

Thesis directed by Professor Gerhard Fischer

Abstract

Although the computer has been viewed as an aid to communication and design, both in face-to-face meetings and asynchronous interaction, collaborative design has proven difficult to support. Designers need to communicate about designs, access and interpret that communication, and understand the feedback that the artifacts and collaborators provide.

Embedded communication clarifies tacit design knowledge by associating communication with the configuration which served to elicit it. Embedded history increases shared understanding between designers by allowing them to recreate the process by which an artifact evolved to its current state.

A prototype system (INDY) instantiates this framework in the domain of local area network design. Using INDY, network designers embed textual and graphical annotations in the design artifact and their design changes are automatically archived. Evaluation of the system confirmed the role that embedded communication and history play in collaborative design and motivated further improvements in the implementation.

Acknowledgements

Gerhard Fischer has thought long and hard about cooperative problem solving and has formed a conceptual framework which respects the user as a person with creativity and individual freedom. Clayton Lewis can always tease out interesting questions to ask about observations and approaches to hard problems in HCI. Ray McCall served as argumentation guide and has a great way with words. Skip Ellis searched for the ''meat'' and reminded me that it is the people who count in groupware. Thanks to Jim Martin for discussions and guidance in the written work. Thanks to Wayne Citrin for exposing me to visual programming and pen-based systems. Thanks to Johnnie Baker, at Kent State University, for introducing me to expert systems.

Thanks to Tom Mastaglio for being a great office mate, co-author, and travel companion. He introduced me to the Lisp Machine and encouraged me to write. Thanks to the Lisp gurus: Andreas Lemke, Andreas Girgensohn, and Alex Repenning for their super help. The dissertation reading group was outstanding in providing constructive help: Kumiyo Nakakoji, Tammy Sumner, Gerry Stahl, and Jonathan Ostwald. Kumiyo was a hard-working office mate who kept a positive attitude in stressful times. Thanks to the network gurus: Hal Eden, Barb Dyker, Evi and Laszlo Nemeth, and Bela Kun, for participating in user studies and critiquing INDY. Thanks to Francesca Iovine for helping things to run smoothly and especially for arranging travel details. Dotty Foerst is the world's best department secretary.

The Colorado Institute for Artificial Intelligence, and the National Science Foundation supported this work and I thank all the citizens for paying their taxes to support these fine organizations. Thanks to NYNEX for sponsoring a fellowship, and especially Ed Thomas, John Thomas, and Mike Atwood.

Thanks to special friends we met here in Boulder. Steve and Debbie Gampp, along with Mike, Heather, ''Gampp Josh'' and Beth were examples of what encouraging is all about.

Betty Balfour took good care of us, and I appreciate her support. Thanks to my parents, Clifford and Mary Reeves, and Wyman Balfour and, yes, even my mother-in-law, Shirley. It could not, and would not, have been done without their support.

Most of all I thank Beth — a true friend and gift. Joshua, Luke, Benjamin and little ''Micah bear'' have a truly unique mother in this world. I get the shingle, but undoubtedly she deserves the credit.

Table of Contents

List of Figures

List of Tables

1. Introduction

This thesis explores the use of computational media in supporting design teams. Designers need to communicate about designs, access and interpret that communication and understand the feedback that the artifacts and collaborators provide.

Two observations provide the high-level guide. First, Rittel observed that buildings do not speak for themselves [Rittel 84]. How can design media do better than buildings at providing their rationale and divulging the process which brought the artifact to its current state? The second observation is by Schoen [1983], who argues that design is a "reflective conversation with the materials of a situation." How can computational media support this conversation and at the same time support collaboration with other designers?

These two questions guide the inquiry and implementation of a computer-based design environment called INDY[1], which supports the collaborative design of local area networks.

1.1 Conceptual Framework

Three resources serve as foundation for the conceptual framework of this research. First is Schoen's theory on design and the metaphor of design as conversation [Schoen 83; Schoen 92]. His detailed analysis of the actions of designers and the role of design materials provides a view of design as a fluid interaction with design artifacts.

The second resource is the work on cooperative problem solving (CPS) and the resulting systems known as domain oriented design environments which brings the views of design as drawing, constructing, and arguing issues [Fischer 92a; Fischer, Nakakoji 91]. Experience has shown the need for allowing designers to use domain-specific abstractions, and design environments should therefore support human problem domain communication.

The third perspective is computer supported cooperative work (CSCW) which emphasizes that people design with people over time [Greif 88]. As described in more detail later, there is much potential in computer support of asynchronous collaboration. The role of context and materials in the design process motivates exploring embedded communication and embedded history as design support tools.

Figure 1-1 shows the goal of this work and illustrates two factors. First, in current computer systems, though designers may use computer-based tools to carry out design tasks, almost all the communication about the design artifact is "offline." It takes place outside the medium in which design is done. The goal is therefore to move much of the communication between users into the human-machine channel. This is indicated by curving the arcs of communication inward to show that they are nearer the computer-based design system. The fact that the arcs do not cross entirely into the computer icon indicates that there will always be communication to which the computer system has no access. The goal is not to restrict communication to only one channel, but to build computer systems that support communication so well that most of it flows through the computational media.

Second, if communication is channelled through the computer system, then the view of that system changes from "keeper of the artifact," i.e. repository of static information, to "design and communication mediator." The larger size of the computer icon in the second picture shows the increased role the computer plays in mediating discussion about design as well as the design work itself.

1.2 Embedded Communication

A review of design theory [Rittel 72; Rittel 84; Schoen 83], cooperative problem solving [Fischer 90], and group work [Grudin 88; Grudin 92; Reeves, Shipman 92a], suggested that one way to provide computer support for design is through *embedded communication*, the goal of which is to embed communication about an artifact *in* the artifact itself.

---

[1] INDY sounds like ND, which stands for *N*etwork *D*esign

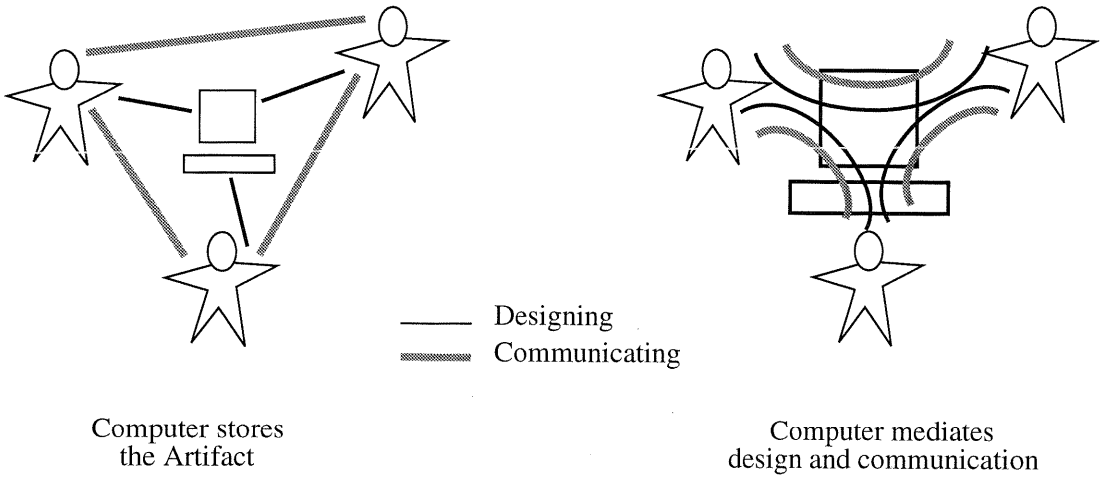


Figure 1-1: Dissertation Focus

The left figure shows how current design environments force most communication to take place outside of the computer system. In contrast, the right view shows 1) how more communication is channeled through the computational design medium, and 2) the computer plays a larger role as medium.

The view of collaborative design as *communication over time* motivates the approach to design media taken here: by moving more of the designers' communication through the computational design medium, the system can integrate *doing* design and the discussion *about* the design.

The computer is not viewed as a device which contains the artifact, but rather as a ''design medium'' which mediates the work of- as well as the discussion about design.

Design materials anchor and elicit communication about the design [Schoen 92]. Talking *about* a design requires talking *with* a design. Whenever possible, collaborators will annotate the artifact directly to communicate critiques and clarify ideas. Computational media provide the opportunity to leverage this tendency with additional support not available in physical design artifacts. For example, physical limitiations due to relative sizes of various design artifacts and, say, textual documents like requirements specifications, makes their integration in the physical world difficult. Whereas computers can easily integrate text and graphics in such a way that they do not physically interfer with each other, e.g. hypermedia documents.

This thesis argues that the best place to keep documentation or design rationale about an artifact is within the artifact itself. This situates the communication in the context in which it was elicited and so helps to expose tacit knowledge hidden in the design artifact. Two aspects of this hidden knowledge are that designers know more than they can say (Polanyi [1966]), and that many references in communication are unclear apart from the context in which the communication took place. Keeping the communication within the artifact aids reader comprehension by clarifying deictic references to parts of the design artifact. Finally, embedding rationale in the artifact itself overcomes the problem of specification and implementation not matching due to the fact that over time they have evolved separately.

Moving more of the communication between collaborating designers through the human-machine channel leads to several benefits. First, it allows communication to take place in the context in which it was elicited. The context plays a crucial role in evoking and anchoring communication about the design. Second, it clarifies deictic references in the communication. The references are clearer when situated in the context to which they refer. This approach is motivated by the key role that tacit knowledge plays in collaborative design [Stahl 92]. Rather than forcing designers to make things explicit which currently go ''unspoken'', the design media should support designers' use of tacit knowledge in the collaborative process.

1.3 Embedded History

Long-term and complex design projects face several barriers to effective collaboration. One is turnover and the loss of the person's knowledge of the problem. Another is that participants have various domains of expertise and as project size and complexity increases, it becomes harder for any one person to know the whole design. Communication issues play a larger role in complex projects [Curtis, Krasner, Iscoe 88]. Decisions overlap and interact. Managing these conflicting decisions is difficult since the designer must be made aware of potential conflicts with his current work.

The approach taken here to alleviate these barriers is through *embedded history*, the goal of which is to embed in an artifact its history. This helps users understand the current situation by supporting an analysis of how the artifact evolved to its current state. History provides memory cues to aid in recognition, lessening the need for recall [Anderson 85].

History also shows the process by which the current design evolved. In domains where visual patterns play a minor role, such as textual representation in software engineering, visual changes in the artifact carry little semantic meaning. However in application domains where visual representations do play a large part, this can be important, as visual changes do carry much semantic meaning.

Table 1-1: Embedded Communication and Embedded History

	Communication	**History**
Theory	Design includes communicating	Design takes place in historical context
Media should support	Integrating communicating and designing	Long-term asynchronous collaboration
Why Embedded?	Provides situated context for communication	Provides historical context for understanding
Mechanism	Textual and Graphical Annotation	Traceable Objects
Immediate Benefits	Context triggers and grounds communication	Unrestrained exploration; What-if scenarios
Long-term Benefits	Clarifies tacit knowledge in the artifact	Restore context and process for understanding

Table 1-1 summarizes aspects of supporting communication over time. Embedded communication is supported by textual and graphical annotation. Both are useful for evaluating artifacts and thus in critiquing collaborating designs' work. Notes serve to remind, explain and critique. Graphical annotations serve to clarify references by pointing and grouping.

Embedded history is supported by an interaction language that is intended to mirror the "language of action" [Schoen 83] for a certain domain. As domain-dependent tools support domain abstractions [Fischer, Lemke 88], so the history language supports the design process or tasks. The artifact can be restored to any previous state to show the context of decisions at that time. Since textual and graphical annotations are also tracked historically, one can see how the annotations developed over time.

1.4 INDY: Collaborative Local Area Network Design

Based on previous work on design environments Fischer [1990], Fischer, McCall, Morch [1989], Lemke [1989], Nakakoji [1993], INDY is a computer based environment for local area network design. It supports design as drawing, constructing, and arguing issues, as well as the view of design as *communication over time*.

Figure 1-2 shows a screen image of INDY. This image is taken from a long-term study in which designers collaborated in making changes to an existing network. Drawing support is provided by graphical objects such as line, ellipse, and rectangle. These objects are shown in the lower left corner. The user selects a tool, then uses the mouse in a style similar to MacDraw^{tm} to draw graphics in the work area.

Domain-specific construction is supported via the palette and predefined design objects such as workstations, routers, and various types of cables. The palette is seen at the left of the screen. As in previous design environments, these constructs represent domain specific abstractions and support design by construction. Textual annotation supports collaborating designers in critiquing each others' work and discussing design issues. Several annotations are seen in the work area.

Artifact history can be accessed from any design unit, or the global view seen at the right of the screen. Navigating through the history is supported via design unit traces (an example trace of a workstation shows that it was created on 11/19/92 and most recently copied on 12/9/92) as well as a cassette-deck interface seen at the top right of the screen.

The lower right-hand corner shows the birds-eye view which places the work area in the larger context of the whole design artifact.

1.5 Reading Guide

Chapter 2 describes the conceptual framework in detail. Schoen's description of a professional practitioner as one who's knowledge is seen *in* action serves as foundation on which cooperative problem solving and computer supported cooperative work are interpreted. The latter two fields provide guidance in terms of computational design support and cognitive issues of collaborating designers.

Chapter 3 motivates embedded communication by an analysis of the metaphor of design as a conversation with design materials. Chapter 4 motivates embedded history by analyzing the role of design processes.

Chapter 5 describes several studies done in the context of this work and the evaluation of the first prototype systems.

Chapter 6 describes INDY, a computer based design environment which supports the collaborative design of computer networks. Chapter 7 describes a long-term study done to both guide the implementation as well as evaluate the effectiveness of INDY.

Chapter 8 describes research related to this dissertation. Chapter 9 provides a brief summary and concludes the dissertation.

The contribution of this thesis is the analysis and implementation of a design medium in which communication *about* the artifact takes place *within* that artifact, and in which all interaction is preserved historically.

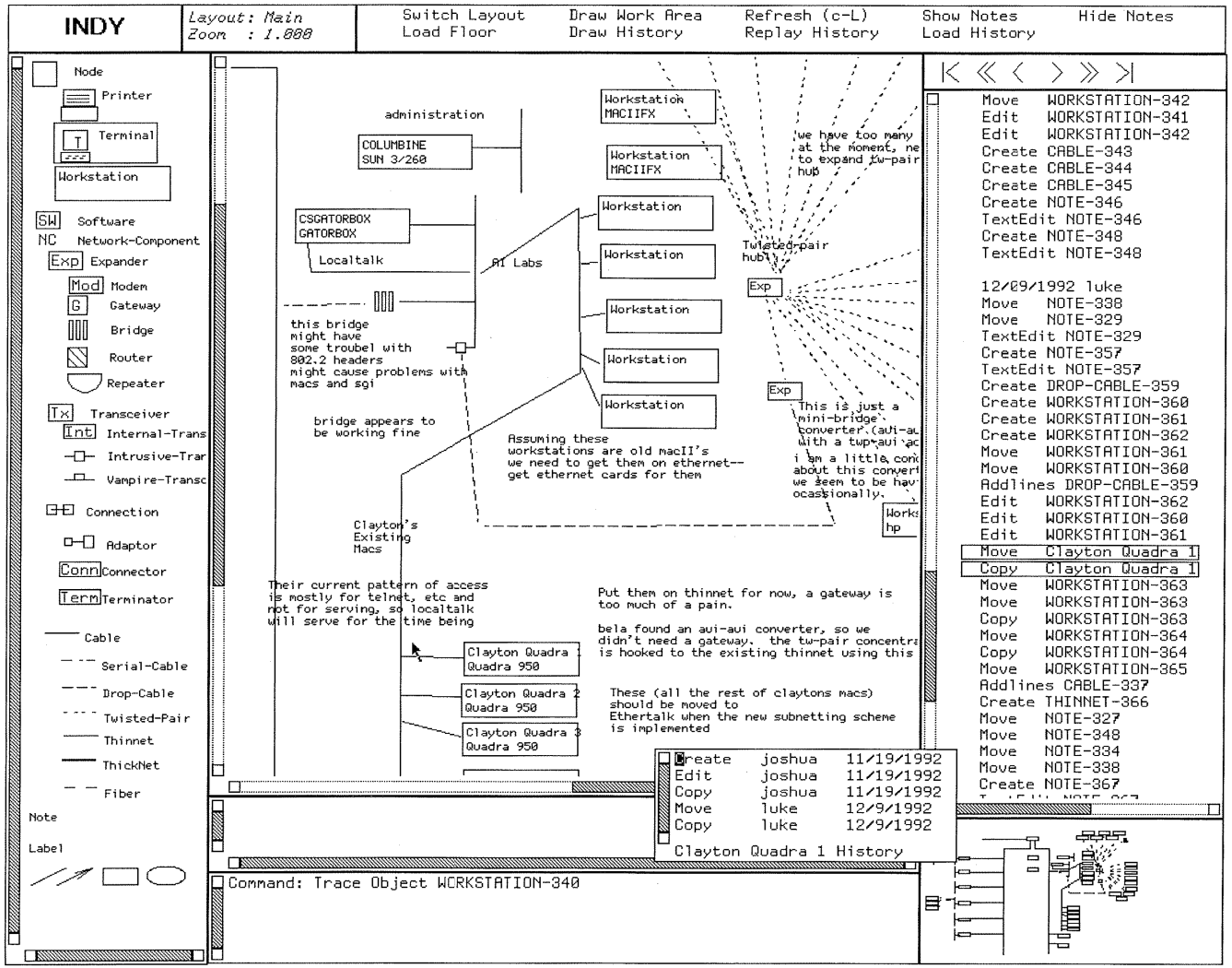


Figure 1-2: INDY: Design Media for Communication over Time

2. Conceptual Framework - Communication over Time

2.1 Problem Statement: Collaborative Design

The general domain of this research is design. In the study of man-made artifacts, Simon observed that "complex systems will evolve from simple systems much more rapidly if there are stable intermediate forms than if there are not [Simon 81, p. 209]." He uses the parable of two watchmakers, the more successful of the two being the one who first builds stable subsystems which he then assembles into complete watches.

However, most design is rarely carried out by individuals working in isolation. Large software products are designed by software teams. Office buildings are designed by an architectual firm. The question arises then, what are "stable intermediate forms?" Are they simply the subsystems in the artifact that can be handed off to other members of a design effort to integrate?

Stable intermediate forms are more than their instantiation; they represent the information necessary for others to interpret. A team of watchmakers would need a common basis for understanding or some explanation of the intermediate forms to assemble them. Rittel's proposal was to encapsulate the design knowledge into a design argumentation. Thus the need for communicating an understanding of the intermediate forms is established.

The watchmakers parable must be extended to include the issue of communication among designers. Designers need to communicate about designs, access and interpret that communication and understand the feedback that the artifacts and collaborators provide.

This chapter describes three resources from which my approach to supporting collaborative design is drawn. They are 1) Schoen's analysis of theory in practice [Schoen 83; Schoen 92], 2) Cooperative Problem Solving Systems [Fischer 1990], [LemkeFischer 1990], [Fischer 1992a], and 3) Computer Supported Cooperative Work (CSCW) [Greif 88]. These resources provide a framework in which collaborative design is viewed as *Communication over Time*, and is thus best supported via computational media which mediate both the design of a product and the communication about that product.

2.2 Design Theory in Practice

This section describes Schoen's [1983] characterization of design as a conversation with design materials. This view serves as design theory foundation of the conceptual framework.

Schoen [1983] investigates theory in practice, and argues that a competent practitioner, a professional, does not distinguish clearly between the theoretical and practical, but integrates theory *in* practice.

To illustrate these concepts, Schoen describes in detail several protocols of practitioners exhibiting what he calls knowing-in-action and reflection-in-action. These labels are an attempt to describe the intertwining of knowing and doing, of evaluating even as one designs. One of these protocols is about a design session in which Quist, an architect, reviews the progress of a student, Petra. In introducing the protocol, Schoen describes designing as a "conversation with the materials of the situation" [Schoen 83 p. 78]. Conversations are give-and-take, back-and-forth; the label "conversation" carries with it expectations of timing. Human conversations move quickly and sometimes lead to unanticipated results or breakdowns which need repair. In further describing this "conversation" as playing out consequences, he writes:

> Each move has consequences described and evaluated in terms drawn from one or more design domains. Each has implications binding on later moves. And each creates new problems to be described and solved. Quist designs by spinning out a web of moves, consequences, implications, appreciations, and further moves [Schoen 83 p. 94]

Notice here the word *spinning* and the time dimension implied. This is not a long drawn out process, but a sometimes rapid-fire activity which shows knowledge-*in*-action, not contemplation *before* action. The rate at which expertise is shown in the interactions causes Schoen to place knowledge-*in*-action.

Schoen thus presents design not as a series of slowly deliberated or carefully analyzed and thought out steps, but as a fluid interaction with the materials of the situation. Design proceeds in small, incremental steps and each partial move puts the whole design at stake.

In keeping with the metaphor of ''conversation,'' Schoen also describes the ''language of doing,'' made up of the parallel ways of designing, namely drawing and talking [Schoen 83 p. 80]. It is this combination of drawing and talking which motivates the emphasis in INDY on providing a design medium which integrates *doing* design and *talking about* it.

Simon similarly describes design as a process in which the designer uses materials which in turn influence the design:

> The emerging design is itself incorporated in a set of external memory structures: sketches, floorplans, drawings of utility systems, and so on. At each stage in the design process, the partial design reflected in these documents servbes as a major stimulus for suggesting to the designer what he should attend to next [Simon 81 p. 109].

Closely related to this, but motivated rather by inadequacies of planning research, Suchman [1987] points out difficulties encountered when systems are fielded which are based on the idea that people operate according to plans. Suchman's critique of planning research is that it overlooks the fact that plans are not the driving force in problem solving, but ''merely'' one of many resources. Plans are a resource, and the ''course taken is contingent on unique circumstances that cannot be anticipated in advance'' [Suchman 87].

The view of design for the conceptual framework is thus the metaphor of designing as a conversation with the materials in which knowledge is seen *in* action. This metaphor agrees with the observation that competent practitioners usually know more than they can say [Polanyi 66], since conversation also leaves many things tacit.

One could attempt to overcome these tacit aspects by forcing designers to make more knowledge explicit. Against this approach, which might be labelled the ''tyranny of the explicit'' [Hill 89], these aspects are seen as motivation for providing computational media in which the designer's natural level of tacit knowledge is respected. The design process suggests the need for a medium in which the design artifact emerges, and which allows the designer to undergo frequent ''shifts in stance'' [Schoen 83 p. 101].

Based on this view of media, the goal is not to automate as much as possible, or to build artificial intelligence into the design media, but to explore basic ways that computational media can increase the effectiveness with which designers engage in design and produce design artifacts [Fischer, Nakakoji 92].

2.3 Domain Oriented Design Environments

The second major aspect of this conceptual framework is the research done on domain oriented design environments [Fischer 92a]. Two aspects of that work are important here. The first is that people experience problems as open-ended and ill-structured [Rittel 72; Fischer, Nakakoji 92; Nakakoji 93]. Individuals arrive at solutions by iteratively reframing the problem [Lave 88; Rittel 84]. Design environments should support problem finding as well as problem solving.

The second facet of the design environment view that serves to guide this work is the emphasis on empowering designers rather than automating design [Engelbart, English 68; Woods 86; Fischer 90; Lemke 90]. Computer systems are seen as an opportunity to help users solve open-ended design problems, and this view leads to an interest in cognitive issues of humans more so than on representational issues of computer systems. Computational systems present an opportunity to provide designers with access to information which helps explain the process and rationale of design artifacts. Viewing communication of design rationale as an information access problem led to studies and analyses of problem solving and design which resulted in the formulation of several challenges related to computer support [Fischer, Reeves 92]:

1. users do not know about the *existence* of information,

2. users do not know how to *access* information,

3. users to not know *when* to use information,

4. users cannot *combine, adapt, and modify* information according to their *specific* needs.

This research address the first two challenges. As design materials, such as the logical diagram shown in the screen image in Figure 1-2, are exchanged among members of a design team, they are marked up with individual's comments. Similarly, drafts of papers are marked up by different coauthors. The objects in the design become the key to indexing comments for future retrieval.

2.4 Computer Supported Cooperative Work

The primary aspects taken from the Computer Supported Cooperative Work field (CSCW) are the framework represented by the time×place matrix in Figure 2-1 and the focus on people communicating with people. Asynchronous communication is not just a backup for synchronous communication, but is in many cases preferred. Two examples illustrate the benefits of technologically supported asynchronous communication: voice mail and electronic mail.

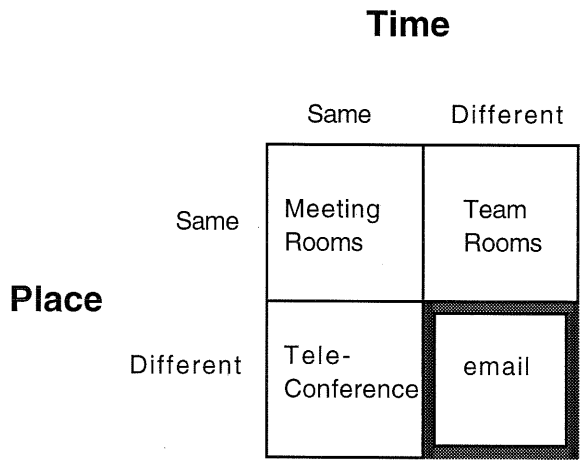


Figure 2-1: Timeframe studied in this Research

For purposes of this project, ''different time'' can be as little as a few hours or as long as a several days and ''different place'' can be the same building or even office area, just so that the collaborators communicate through the design environment.

Consider the installation of voice mail systems in large corporations. Experience suggests that once people become used to the idea of asynchronous communication by phone, they make better use their phone communication. At first people leave messages like:

Hi, this is Denny, I guess you're out, so call me back.

But after a while, people begin to leave messages that contain more than just, ''call me back.'' For example:

Hi, this is Denny. I've found a problem with the AR 30 report. The due dates are incorrect for customer number 899902. We need to get these right before Friday's close.

Though it takes time, people learn that it is more efficient to place an item on another person's ''electronic stack'' or ''inbox'' than to interrupt what they were doing with a phone call. And though people usually prefer to speak to someone in person, they nevertheless learn how to make good use of the technology. Leaving a detailed message makes more sense for many of the messages and requests. It can be better to give the person time to research the question and call back, rather than surprise him with a problem and expect instant diagnosis. Asynchronous communication begins to be used where

synchronous previously dominated. Though phone mail is at first thought of as an inconvenient backup to the more preferable voice-to-voice, it becomes a useful service in its own right, and more than just a back-up.

In arguing against the assumption that the goal of computational media should be to emulate fact-to-face communication, Hollan and Stornetta [1992] cite email as the "paramount success of computationally-mediated informal communication." The interesting aspect related to the point argued here is the statement:

> It meets our critical litmus test of being used by groups even when in close physical proximity. In fact, in our own experience, it is not uncommon to send email to someone in the next adjacent office, or even someone sharing an office.

Like voice mail, in certain situations, email has changed from a tolerable substitute to a preferred medium.

Electronic mail plays an important role as a success model of computer supported communication. Though much CSCW research focuses on providing media which emulate face-to-face meetings, the success of email is a reminder that there is more to good communication support than emulating face-to-face communication. If the artifact is indeed a good medium for communication, more of the currently synchronous communication will migrate over to asynchronous.

The main reason for the migration to asynchronous modes of communication is the diversity of design teams. Case studies show that they are composed of people with different kinds of expertise in different departments [Curtis, Krasner, Iscoe 88; Grudin, Poltrock 89]. Though the members of these teams are on a *team*, these individual differences imply different work priorities and schedules. Out of logistical necessity then, more communication will take place asynchronously.

As more diverse design teams are assembled, the limit will be the individual life-styles and priorities of team members. This author's experience with expert programmers and analysts suggests that they usually prefer to have a chance to prepare for a problem solving session knowing the questions, than to have a meeting in which questions are raised for the first time and answers expected.

A benefit from asynchronous communication is that it is *archived somewhere*. Whether this is digital voice recording, or electronic mail, it is available for later retrieval. Clearly this does not solve the problem of information retrieval later, but it is a first step.

2.5 Design as Communication over Time

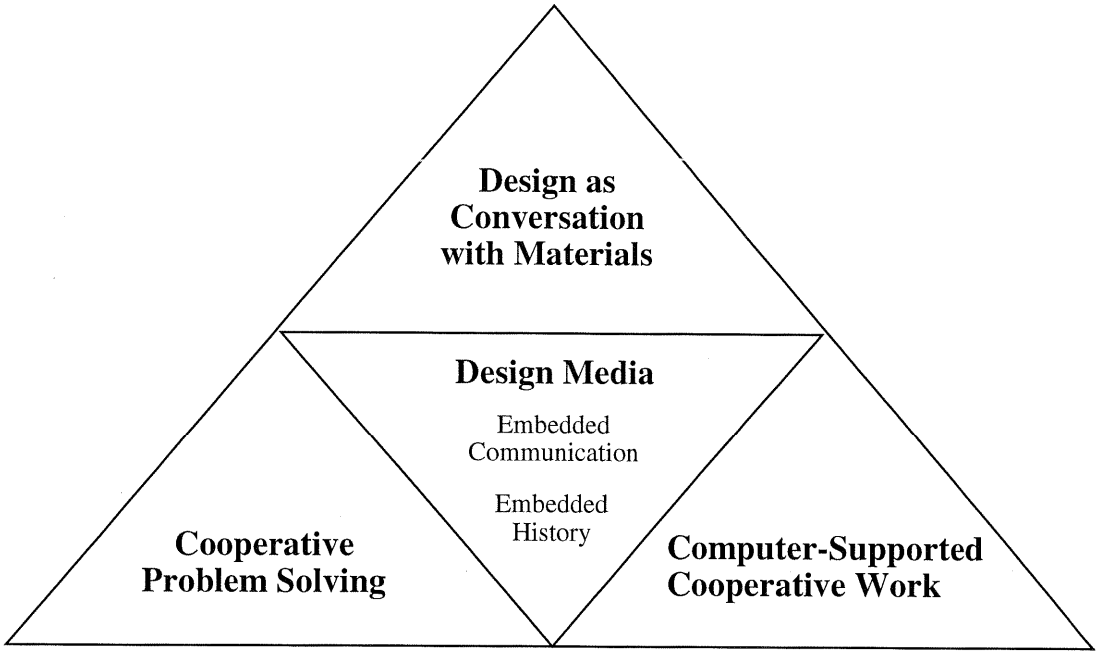
The conceptual framework may now be summarized. Schoen's theory of design provides us with the metaphor of "design as conversation with materials of a situation." Cooperative Problem Solving provide us domain oriented design environments which support drawing, constructing and arguing design issues. Computer supported cooperative work serves to remind us that is people who design and that they do so collaboratively over time. Figure 2-2 shows how these relate and summarizes the details of contribution from each resource.

Schoen's metaphor is interpreted as follows. In light of collaborative design, the materials are seen not as a second communication partner, but rather as the vehicle through which communication is transmitted to another human. Computational media provide an opportunity to support this human-materials conversation and at the same time support collaboration with other designers.

Schoen's materials are static: paper and pencil and tracing paper. In this research, the materials provide a dynamic aspect. One way to exploit this dynamic aspect is by actively revealing constraints [Borning 79; Steele 80], GrossBoyd1991, Fischeretal.1991b]. The dynamic aspect explored in this research is history. A log of all design changes is made for later replay and exploration.

Schoen's situation is the real world context. This work extends this context to include the historical situation. Table 2-1 compares these views.

"Communication" over time. Teamwork is playing a larger role in design projects [Hackman, Kaplan 74; Johansen 88; DeMarco, Lister 87], and this presents special opportunities for computational support



Aspect	Design Theory	Example Systems
Schoen	Conversation with Materials	paper and pencil
Cooperative Problem Solving	draw, construct, argue human problem domain communication	domain oriented design environments
Computer Supported Cooperative Work	people communicating with people over time	electronic mail voice mail

Figure 2-2: Building Blocks

This work rest on three major resources: 1) Schoen's design theory contributes the metaphor ''conversation with materials'' for the design process, 2) Cooperative Problem Solving, and 3) Computer Supported Cooperative Work (CSCW). The result is a view of design as collaboration through a medium over time.

Table 2-1: Extending Schoen's Design Metaphor to Collaborative Design

	Schoen	INDY
Conversation	Designer and Materials	Designer and Designer via Materials
Materials	Static	Dynamic
Situation	Real World Context	also Historical Context

of communication. Even when it is possible for collaborating designers to have face-to-face meetings, there is still much potential in providing tools primarily intended for asynchronous use [Hollan, Stornetta 92]. Though synchronous communication via face-to-face meetings or video tools is certainly important in design, the approach taken here is to focus on the aspects of design which can be enhanced by asynchronous support.

Many group communication systems exist [Winograd 1988] [Maloneetal.1986] [Maloneetal.1988] [ShepherdMayerKuchinsky 1990] [HalaszMoranTrigg 1987], but as mentioned above, the CSCW community generally acknowledges that the only groupware to really have succeeded is electronic mail [Hollan, Stornetta 92; Grudin 92]. Two prominent features of email are unstructured text and the asynchronous time-frame. Based on email's somewhat exclusive claim to success, systems which enforce no structure on communication seem to have the most promise.

Communication "over time". Much research in supporting collaborative work has gone toward supporting synchronous communication, e.g. COLAB [Stefik et al. 87] and GROVE [Ellis, Gibbs, Rein 91]. However two aspects of collaborative design motivate research on long-term asynchronous communication. First, direct communication is not possible when the designer responsible for a previous design decision is no longer with the project or the company. Studies by the National Institute of Standards and Technology and the U.S. Air Force have found that up to 75 percent of the total maintenance effort is enhancement in complex systems [CSTB 90]. Communication support must therefore not only help during the "design" phase, but over the whole life-cycle of the project. The original design team will rarely stay the same over the entire life-cycle. Artifact history serves not only the current designers, but those in the future who need to understand the design they are responsible for changing.

As project length increases, human memory limitations begin to play a role in the design process [Anderson 85]. It thus becomes important to provide external memory support. Computational media should be provided to help keep designers from inferring design decision rationale based only on plausible inference or interpreting current design aspects divorced from their historical context [Reder 82]. An embedded design history does this by embedding the historical context and the communication where they are most useful, in the artifact itself.

In the same way that evidence of physical history guides cognitive tasks, computational media should provide cues of use which guide design tasks [Hill et al. 92]. Computational media represent the potential to provide history access mechanisms which go beyond what is possible with physical artifacts, such as providing access by various perspectives such as date, user, design change, and relation to other design units.

2.6 Embedding Communication and History in Design Environments

Computer systems for design have placed an artifical separation between *doing* design and *communicating* about design. The extremes are at one end an exclusive focus on the *product*, which raises questions about computational representation, and at the other end an exclusive focus on the *process* of design, which raises questions about how design takes place and the role of the social context. Each focus informs the construction of computational systems intended to support design, but a synthesis of these two extremes is clearly beneficial. Though collaborating designers do communicate in social settings, they communicate *about* a design product; and though designers may do much of their work alone, communication nevertheless plays a key role in their design work.

If collaborative design is seen from the perspective of communication over time, an ideal design support system would provide a medium in which collaborating users could *do* design and *talk about* that design. As motivated in more detail in Chapter 3, the communication would be integrated in the artifact in such a way as to provide access to communication in the context in which it is most relevant. As motivated in more detail in Chapter 4 the medium would also store the history of the artifact to help designers understand previous decisions and learn from previous examples.

To meet these goals of a design medium, INDY builds on the work done on design environments by supporting design as *drawing*, *constructing*, and *arguing issues*. It goes beyond these views by supporting the view of design as communicating over time. INDY provides a medium in which communication and the process of design are embedded.

2.6.1 Computer Support for Design

Research on computational support for design illustrates how one's view of design influences the type of computational support suggested. This research builds on the following views of design: as drawing, constructing, and arguing viewpoints.

Design as drawing. Drafting tools illustrate aspects of viewing design primarily as *drawing*. The designer draws using geometric constructs such as lines, ellipses and rectangles. Tools such as MacDraw[tm] and MacPaint[tm] illustrate tradeoffs between freeform drawing (bitmaps) and manipulating geometrical objects (objects in MacDraw[tm]). Tools such as Freehand[tm] illustrate how computers can provide sophisticated curve-drawing support such as easy manipulation of Bezier curves. These tools are domain independent in that no special support is provided for any one application domain, except perhaps the drawing domain itself. First generation CAD systems provided this level of drafting support [Ullman 91; Balachandran, Rosenman, Gero 91].

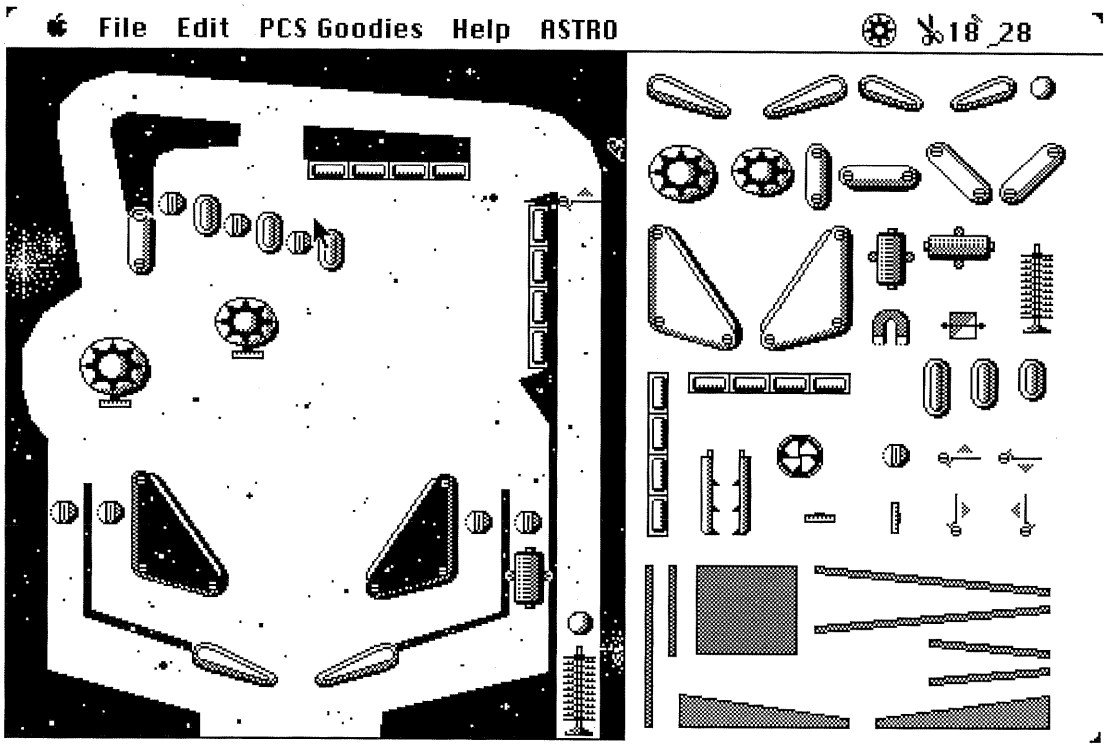


Figure 2-3: Pinball Construction Kit

Domain-specific constructs (shown on the right) bridge the distance between a designer's abstractions and what the system supports. They can also encode behaviour which in this case is used to simulate the constructed design (shown on the left).

Design as construction. The extreme difficulty for computational media to recognize collections of geometrical shapes as higher-order semantical units in combination with the potential to provide design support based on semantical representations lead to domain specific construction kits [Budge 83; Fischer, Lemke 88; Gance 90]. CAD systems now go beyond drafting and support general and domain-specific modelling [Ullman 91; Balachandran, Rosenman, Gero 91]. Computer construction kits such as the Pinball Construction Kit [Budge 83] (illustrated in Figure 2-3) provide application-specific abstractions such as bumpers, flippers, and counters which the user selects and arranges via direct manipulation. For any given domain, an analysis is made of the useful design abstractions and those are provided as building blocks from which a design is constructed.

Application-specific constructs shorten the distance between a designer's vocabulary of design and the functionality provided by the system. This kind of communication between the computer and the user is called human problem-domain communication [Fischer, Lemke 88]. Computational support can be provided at a higher level than *drawing* tools, since behavior can be attributed to higher-level abstractions. One example of this support is that in the Pinball Construction Kit, playing the constructed design is in essence simulating the design.

Design as argumentation. Systems such as GIBIS [Conklin, Begeman 87] [Conklin, Begeman 88] and PHI [McCall 91] are motivated by a view of design as *argumentation*. Computational support is intended to simplify constructing and evaluating conflicting issues in design. In recognition that design is open-ended [Rittel 84], the focus is on considering alternative perspectives. Though the Issue Based Information System (IBIS) was initially developed as a methodology separate from the construction of a design artifact, systems such as JANUS have viewed computational media as an opportunity to integrate construction and argumentation [Fischer, McCall, Morch 89b]. Figure 2-4 shows an example of the role of an argumentation component.

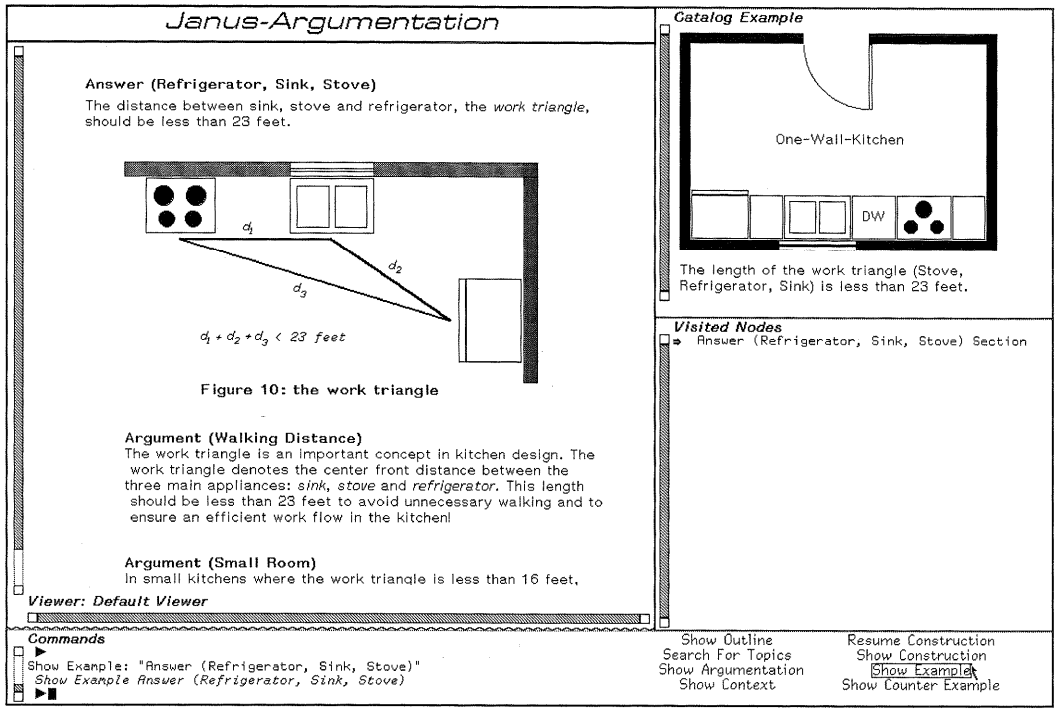


Figure 2-4: JANUS-ARGUMENTATION: Rationale for the Work Triangle Rule

JANUS-ARGUMENTATION is an argumentative hypertext system. The *Viewer* pane shows a diagram illustrating the work triangle concept and arguments for and against the work triangle answer. The top right pane shows an example illustrating this answer. The *Visited Nodes* pane lists in sequential order the previously visited argumentation topics. By clicking with the mouse on one of these items, or on any bold or italicized item in the argumentation text itself, the user can navigate to related issues, answers, and arguments.

Design Environments. The systems which most guided the implementation of INDY are toolkits [Budge 83] and design environments [Fischer 92a; Fischer, Nakakoji 91].

Construction kits and human problem-domain communication are necessary but not sufficient for good design. Upon evaluating prototypical construction kits [Fischer, Lemke 88], it was seen that they do not by themselves assist the user in constructing interesting and useful artifacts in the application domain. To do this, systems need knowledge to distinguish "good" designs from "bad" designs. Design environments combine construction kits with *critics* [Fischer et al. 91a]. Critics use knowledge of design principles for the detection of suboptimal solutions constructed by the designer. They were seen as one way computational media can provide the "back-talk" implied in Schoen's metaphor of design as conversation. One of the challenges for critiquing systems is avoiding work disruption. Design environments can accomplish this by making the critics sensitive to the specific design situation [Nakakoji 93], incorporating a model of the user [Fischer et al. 91a], and giving users control over when and which critics should fire [Fischer, Girgensohn 90]. JANUS [Fischer, McCall, Morch 89b] has demonstrated that critics do not need to be highly intelligent to be useful. Quite simple critics can be informative because they are based on domain knowledge that designers might not have (e.g., network designers are not necessarily familiar with relevant knowledge about fire codes for buildings).

Table 2-2 summarizes the computational support suggested by these views of design as drawing, constructing, and arguing issues. This research builds on these foundations by focusing on embedding communication and history in a design medium. The dimension of *time* is supported by providing a design medium which transparently archives all changes.

Table 2-2: Views of Design

Design as	Computational Support
Drawing	low level graphics such as line, ellipse, rectangle
Construction	higher level sketching support via domain specific abstractions
Argumentation	construct and evaluate perspectives on issues; critiquing support to raise issues
Communication over time	textual and graphical annotation; design artifact history

2.6.2 Embedded Communication

INDY supports embedded communication by integrating textual and graphical annotations in the artifact under construction. The operations for manipulating text and graphics are the same as other design units such as bridges and workstations; for example scaling or hiding.

Previous efforts to connect design and discussion have integrated construction and argumentation [Fischer, McCall, Morch 89a]. A hypermedia database was linked to the construction situation via critic rules. When a constraint was violated for which there existed an issue in the issue-base, the user would be positioned at that place in the issue-base.

Linking argumentation to construction was a step in the right direction and the approach in this dissertation extends this by going beyond linking to embedding. Rather than linking the construction situation to a static argumentation base, a medium is created in which communicating and designing both take place. Collaborating designers critique a common artifact and use the artifact as the medium through which argumentation is communicated.

2.6.3 Embedded History

Users draw heavily from past experience in solving current problems [Lee 92a]. Computational tools should therefore support this human tendency to reuse previous experience. However, a complicating factor is the tendency for people to ''misremember'' an event according to plausible inference rather than exact recall [Reder 82].

The reason history is important in long-term design projects is that memory performance improves the more closely the current context matches the past physical, emotional and internal context [Anderson 85]. Restoring the context around which a past decision was made can aid in helping understanding. The process by which information is created and used can be important for understanding of the end product of the work group [Wolf, Rhyne 92]. So the history state as well as the history process is important in helping designers understand past states of the artifact and thus past rationale for decisions.

The approach here is to recognize the fluid nature of the design process and create a computer-based environment in which the artifact is less of an end-product and more of a process. If capturing the design process can be done in a way that does not interfere, then others will be better able to learn from observing the design sequence later. Understanding a complex design is best done by studying the process as well as the product [Kuffner, Ullman 91].

2.7 Summary

The view of collaborative design as communication over time is founded on three resources. First is Schoen's theory on design and the metaphor of a conversation with materials of a situation. Second is the CPS perspecive which resulted in computer-based design environments and which brings the views of design as drawing, constructing, and arguing issues. Third is the CSCW perspective which emphasises the role of collaborating and communicating over time. The role of context and materials in the design process motivate exploring embedded communication and embedded history as design support tools.

3. Embedded Communication

This bottle was in a sealed carton. Do not use if carton was open or damaged.
Warning note printed on a bottle of contact lens cleaner

The note referred to in the quotation is an example of embedding communication about an artifact in the artifact, described in detail in this chapter. Another example from the same domain, contact lens care products, illustrates the problem with *not* associating communication closely with the artifact. Enzymatic cleaning tablets come in various strengths. A certain brand packages them in a box with the following contents:

- clear plastic cleaning viles
- folded sheet of (contra)indications
- cleaning tablets wrapped in foil, each tablet marked as follows:

```
Obverse:                Reverse:
  ENZYMATIC               Distributed by
   Cleaner                 Store-X
   (papain)                LOT  NNNNT
                           EXP  Jan.95
```

The problem with this arrangement is that the indication of strengths of tablets is not on the tablets themselves, but on the folded sheet of paper. In actual use, if one buys different strength tablets, one must keep the sheets and tablets together, because the sheet for the ''slow'' tablets states:

> Soak the lenses... overnight (a *minimum of 8 hours*).

whereas the sheet for the ''fast'' tablets reads:

> Do not soak high water content (55% or more) contact lenses longer than 2 hours, otherwise ocular irritation may result. Should ocular irritation occur, immediately remove your contact lenses, clean and disinfect the lenses again and reapply. If irritation continues, remove your lenses and consult your eye care practitioner.

However it is tedious to have to keep the sheets; a simple label on the tablet indicating strength would suffice. *Fast*, *Slow*, *Extra Strength*, *Regular strength*, *2 hour maximum*, *Minimum 8 hours* etc. are examples of all it would take to remind the user. The more new types of tablets with different time constraints and risks are introduced, the more important it is to describe the product *through* the product.

This small example motivates *embedded communication* in the physical world. The computational world of electronic media provides an opportunity to do this at a larger and more complex scale.

3.1 Introduction

The objective of *embedded communication* is to: *embed within an artifact the communication about it.* Rather than being just an instantiation of a product idea, the artifact serves as medium through which communication is channeled during the design process. This chapter argues for *embedded communication* by analyzing the nature of design artifacts and the potential for computational-based medium to support new kinds of interaction. This chapter revisits Schoen's description of design [1983, 1992] presented in Chapter 2 in more detail to more specifically motivate INDY's approach to *embedded communication.*

3.2 Motivation for Embedded Communication

The motivation for this idea is taken primarily from Schoen's [1983, 1992] design theory, along with observations of design sessions of local area network designers. As motivation for *embedded communication*, two points are considered. First, observations of collaborating designers show that artifacts serve as medium for communication. Second, discussions about the artifact guide the incremental design process.

3.2.1 The design artifact as medium for communication

An important aspect of design work is that the design artifact plays a crucial role in grounding discussions about the artifact. Schoen's [1983] describes a designer as follows:

> He shapes the situation, in accordance with his initial appreciation of it, the situation ''talks back,'' and he responds to the situation's back-talk. (p. 79)

As an architect, for example, works out a design problem, the materials of the situation are more than just passive repositories for his actions. As collaborating designers share common artifacts, they use the materials to communicate, design, and annotate each others' work.

Two aspects of this interaction with the situation are: 1) talking *about* an artifact requires talking *with* that artifact, and 2) annotations are attached to the artifact itself.

Talking *about* an artifact requires talking *with* that artifact. Observation of designers in architecture show that the discussion is grounded so fundamentally in the representation on paper or white-board that the artifact shapes the communication between designers.

A closer look at the protocol of Quist conducting a design review with Petra, mentioned above in Chapter 1, shows the intertwining of drawing and talking. Quist listens to Petra's description of the problem and her difficulty in continuing the design. He places a sheet of tracing paper over the sketches and begins to draw on it. As he draws, he talks. And as he speaks, he draws, placing, for example, the kindergarten ''here'' in the drawing.

> His words do not describe what is already there on the paper but parallel the process by which he makes what is there [Schoen 83 p. 80].

The verbal and non-verbal dimensions are so closely connected that the spoken words are unclear apart from the sketching, and the sketching is unclear apart from the spoken words. Deictic (referring to items with words like *this*, *those*, or using words like *here*, *there*) utterance litters the transcript. Since sketching plays such a central role in architecture, it is easy to dismiss this dependency on deictic references to make sense of the communication as simply related to that domain. But observations of professionals in other domains such as network design shows that this dependence is not restricted to the domain of architecture. An analysis of two network designers planning the installation of new workstations (described in Chapter 5) showed that deixis also plays a key role in that domain. In design work, the communicating and designing both take place within the artifact.

Annotations are attached to the artifact itself. An issue which is closely related to the claim that ''talking *about* requires talking *with*'' is that written comments about an evolving design artifact are not separated from it, but placed *on* or *in* it. The distinction is important to designers of design tools.

To continue with Schoen's design review transcript, when Quist comments on Petra's sketches, he sketches on the tracing paper which is over Petra's work. The meaning of his sketches are only clear in relation to the underlying design. However, each time a comment is made, the person does not add another piece of tracing paper. The first sheet serves as main anchor for other annotation. Yet there is also freedom to add a sheet on top of another sheet and layer the comments and perspectives or start fresh when the changes differ significantly from the existing sketches.

As more and more design domains have moved ''on-line'', the simple ability to sketch on or in the artifact was evidently seen as unnecessary. For example, CAD tools do not emulate the ease of graphically annotating paper and whiteboard. They serve the task of presentation, but not the process of designing.

3.2.2 Critiquing through the Artifact

Research on Cooperative Problem Solving Systems has identified critiquing as a major factor in taking computational media beyond *tool* to *design environment* [Fischeretal 1990], [Fischeretal 1991b], [Lemke 1990b]. The critiquing which INDY tries to support is human-human rather than computer-human. For purposes of this discussion, critiquing is defined as ''considering merits and demerits or evaluating pros and cons.''

In architecture education, the evaluation of one's student work through design reviews and the ''crit'' (an evaluation session in which several designers review a design), plays a major role. But more than that, it is a part of the core discipline. Learning how to become an architect includes learning how to give and take ''crits''.

Architecture. Schoen's design review scenario shows how collaborators use the artifact to mediate critiques. But not only does it mediate the critiques from a distance or merely as visual cue, the artifact provides the means of carrying out the critique. Though some of Quist's comments are high level:

> The principle is that you work simultaneously from the unit and from the total and then go in cycles...

and so are not connected directly in any one sketch or part of the drawing, most are anchored in a part of the drawing with is then modified as the critique is made.

> Now in this direction, that being the gully and that being the hill, that could then be the bridge, which might generate an upper level which could drop down two ways.

As Quist says this, he modifies the drawing by adding directional lines and a rectangle showing the proposed upper level. Though Schoen's scenario summarizes at several points and does not include all the conversation, it becomes clear that whenever there is the opportunity of critiquing via the drawing itself, it is taken. When a suggestion is made which currently has no representation, a representation is quickly made and it then serves to anchor further critiques and refinements. The participants move easily between levels:

> Q: Now this would allow you one private orientation from here and it would generate geometry in this direction. It would be a parallel...
> P: Yes, I'd thought of twenty feet...
> Q: You should begin with a discipline, even if it is arbitrary, because the site is so screwy-- you can always break it open later

Designers will take every opportunity to illustrate a critique. It is the exception when a detailed comment is made and not accompanied by changing the artifact.

Architecture has evolved the concept of scaled drawings so that one can easily evaluate relative distances. Schoen's design review mentions scale:

> P: I had six of these classroom units, but they were too small in scale to do much with. So I changed them to this much more significant layout (the L shapes)... then that opens into your resource library/language thing.
> Q: Is this to scale?
> P: Yes.
> Q: Okay, say we have introduced scale. But in the new setup, what about north-south?

Knowing whether the drawing is to scale is important in Quist's interpreting the design. In a similar way, the basic symbols that have evolved in network design carry much meaning and are easily recognizable. The logical diagram has evolved to communicate connectivity, a primary criteria in networking.

In domains where the primary representation is not graphics, such as collaborative writing, similar abstractions can be found. Since text is visually so different from sketches of architecture and network design, one might expect critiquing interaction to be quite different. But examples of critiqued documents such as draft research papers show that in the same way that sketches ground and serve as vehicles for critiques, so the text document itself also grounds, and serves as vehicle for, critiques.

Network Design. As described in more detail in the Chapter 6, network designers have over time evolved the logical map to serve as a key design document. It abstracts away many physical characteristics of the network but preserves connectedness. Types of devices are represented by icons that were agreed on by consensus of the community of network designers.

The logical map serves not only to represent the real network, but also as medium through which changes are considered and argued. It focuses as well as facilitates discussion. It is frequently in arguing over these documents that specific issues lead to discussions of larger issues. Collaborating designers prefer to ground discussions in design representations. In video-taped sessions, network designers were observed as they explained previous design decisions and solved theoretical and upcoming expansion problems. The logical maps served to:

- point out inconsistencies between an appealing idea and its difficulty of implementation,
- remind participants of important constraints,
- and describe network states before and after changes.

Logical maps serve as media through which critiques are illustrated and argued.

Critiquing in Design. There is an aspect of critiquing that *embedded communication* does not address. It is important to distinguish between suggesting a critique and carrying it out. Suggesting can be accomplished by talking or writing about it, or perhaps simulating the effect of the recommendation. Many critiques are not just spoken, but actively carried out. In Schoen's design review, Quist sometimes makes sketches which serve as critiques. The spoken aspects clarify the intent of the recommendation or change. The same is true in network design, where collaborators will erase parts of white-board sketch and redraw it according to a new insight or critique.

When the face-to-face aspect of this words-and-action critiquing is removed, as in asynchronous communication, one can no longer explain the changes as one does them. One can sketch, and then leave notes explaining the sketches. But a challenge is to honor how people tend to *carry out* their critiques. Computer-based media should allow collaborators to *do* the critique as well as write about it. In Schoen's design review, the introduction of tracing paper allowed Quist to carry out his critiques while preserving Petra's original drawing [Schoen 83]. In the same way, computational media should allow critiques to be acted out within the artifact without worry of changing something permanently.

3.2.3 The Large Role of Small Talk

Schoen's protocol analysis shows how even relatively small- or short-term problems such as an architectural design review involve many small or incremental changes in perspective. Though one can point to important turning points in the design process, Schoen observes how small steps contribute to continual reframing and exploration of the problem and thus result in the larger turning points. The designer goes through frequent ''shifts in stance'' [Schoen 83 p. 101].

According to Schoen's observations, designing and talking about the design are intertwined. Analysis of his protocols shows that discussion and action are of equal importance in the incremental nature of design. It is in this sense that ''small talk'' contributes to design.

If design proceeds in small steps and if discussions are an integral part of the proceedings, then this presents a major challenge for asynchronous collaboration. In an important sense, *small talk* is impossible due to the time lapse between turn-taking. The talk cannot affect the design in the same way it does in face-to-face collaboration.

Partly to address this problem, voice annotation systems are used to allow people to express decision rationale verbally. Though digital voice is storage intensive, hardware technology is advancing at such a rate that the question is not *whether* to keep something, but how to take advantage of the fact that we now do keep it. For example, a large defense contractor keeps digitized voice annotations as design rationale, even though technology is not yet to the point where these digitized voice files can be processed by anything but the human ear [Dehn 93]. The hope is that in the near future, technology will be available to index and search these voice archives.

Therefore one assumption guiding the INDY implementation is that storage of communication via textual annotations will not prove to be the major scaling obstacle. In support of this, consider that the network design group at CU currently archives all email communication related to trouble reports, and keeps separate email diaries by machine. The question is not whether this communication is important enough to keep; that judgement has already been made. The question has to do with providing access to the information so that designers gain the maximum benefit.

3.3 Challenges

The goal of embedding communication in a design artifact is to help designers:

- support the natural intertwining of doing and commenting on design
- understand other's work by having access to the communication that affected the design

Table 3-1 shows a framework for analyzing these challenges. Place and Time (Here and Now) are used to categorize the problems that must be overcome in supporting embedded communication. The primary motivation for viewing challenges in these terms is Grudin's observation that groupware systems will fail if they cause users to do work for which they do not profit [Grudin 89]. Though *embedded communication* is intended to serve long-term asynchronous communication, it must nevertheless serve the designer *here* and *now*.

Table 3-1: Embedded Communication in terms of Here and Now

Context	Motivation	Benefits
Here	artifact elicits communication	associate communication with context in which it was elicited; clarify deictic references
Now	doing and talking are part of the same process	avoids problems of doing now and supposedly describing later

Design artifacts vary in the level at which they represent the product under design. Several types of representations serve to guide the design process. Text design, or collaborative writing, also uses different representations such as outlines, electronic files, and the physical printed result. Text seems particularly well suited to accommodating communication *about* the document *in* the document. Word processor features such as *hidden text*, *comments*, and *change bars* are common and support *embedded communication* [Microsoft 88; Reid, Walker 80]. And the PREP system was motivated by how glosses were originally placed in the margins of manuscripts [Cavalier et al. 91].

In other domains, it is less common to mix *communicating about* and *designing* in the same medium. In architecture scaled models, 2- and 3-dimensional drawings, and blue prints serve as representations. Each of these serves a particular purpose and presents challenges to using the media in which it is represented as a vehicle through which communication flows. Blue prints for example usually have small annotations clarifying directions of staircases, power requirements, and expected locations of equipment such as phone patch panels, closets, etc. connecting the 2d drawings with the expected *use*. But one does not see the discussions or critiques that led up to the various design decision; only their results.

Chalfonte, Fish and Kraut [1991] did an experiment in which participants made either written or spoken annotations to a document to help a fictional co-author revise it. When the subjects' annotations were restricted to writing, they commented on more local problems, whereas spoken comments addressed higher level concerns.

When asked to use written text to express comments on high-level issues, they were "less successful." Thus the challenge is to facilitate the different levels of communicating necessary in critiquing others' work.

3.4 Limits of Embedded Communication

3.4.1 Social Limits

In Schoen's transcript of the design review, Quist placed tracing paper over Petra's design and made annotations on this tracing paper. After the design review he could take the tracing paper and thereby remove all corrections, challenges, and comments. By separating the comments from the artifact, they do not become permanently public.

The reasoning in this chapter has been that since collaborators *want* and *tend* to embed communication *about* the artifact *within* that artifact, computational design media should support and encourage this. Doing this however poses a privacy risk. Whereas in most design situations in which comments and changes are embedded in an artifact, that artifact is usually only a second order representation or a copy of such a representation. Even when the artifact is physical, such as a paper that is being reviewed, and the commentator annotates the physical copy of the paper, it turns out that the *real* paper is in the form of an electronic file.

Embedded communication is not about annotating disposable *copies* of the artifact, but the artifact itself. As will become clearer later in the description of user interaction with INDY, however, *embedded communication* exposes designers to public scrutiny. For example, if you examine the CAD drawings of a building, there is no possibility of finding incriminating evidence in terms of discussions about a certain design tradeoff. Those discussions and any resulting documentation is stored separately. If all communication is embedded in the design artifact, this could be used against the original designers later on, when one discovers that a design decision had tragic implications. When it comes to high-profile design teams, there is security in numbers - that is social customs have developed with respect to committees, so that the committee, and not a given individual, takes responsibility for a decision. Whether this is good is not the issue I address. What is at stake, though, is that a technology like embedded communication can short-circuit social conventions like committees when it comes to hiding responsibility.

One could certainly make communication anonymous, but this ignores how collaborators take advantage of knowing with whom they are dealing. So while providing for anonymity is easy enough technically, it may get in the way of the very process it is trying to promote.

3.4.2 Technical Limits

The most obvious limitation of *embedded communication* is that of storage space. INDY makes the simplifying assumption that textual annotations in some sense suffice as ''communication,'' but an ideal instantiation of this would require a medium in which more forms of communication could be stored. Video tape meets this criteria, yet is not a part of current collaborative design. The issue is not how much data can one store, but how *useful* it is. Evidently current video technology is not deemed useful enough to be in widespread use as design-enhancing technology. It is not the quantity of information that is the scaling limit, but the lack of tools to index the relevant information and make it accessible to current and future designers.

3.5 Summary

The goals of *embedded communication* are to help users talk about, critique, and understand other's work, as well as overcome certain inherent limitations of synchronous collaboration. It serves designers by providing support *here* and *now*. The *here* acknowledges the importance of the designer's place in a situated context [Schoen 83; Suchman 87]. The context elicits communication and should therefore be associated with it. The *now* acknowledges that *doing* design includes *communicating* about that design.

4. Embedded History

History tools serve several needs in human computer interaction. Lee [1992] argues that they can help alleviate problems in human computer interaction such as variability from user to user and session to session. User activities contain repetition, which motivates a way to provide support for easily reusing previous commands [GreenbergWitten 88]. Studies of the use of history have focussed on history as a domain- and application-system independent tool for user support [Lee 92b; Barnes, Bovey 86]. The objective of *embedded history* is to *embed within an artifact its history.*

Though repetition plays a part in human-computer interaction, it is only a small part. History tools have far greater potential than command-line recall. Some of the potential relates to the role that context plays in human problem solving. There is much potential for computer systems to serve as external memory aids in restoring the context surrounding past design decisions [Suchman 87; Anderson 85; Reder 82]. The context becomes all the more important as collaboration increases. In the context of collaborative design, it is not enough to provide *user* history, there should be *artifact* history.

4.1 Motivation

4.1.1 Human Computer Interaction

Computer tools which incorporate history are based on the observation of how humans use the past to solve current problems [Lee 92a]. The common use of a history tool is to reuse and possibly modify a history item to save keystrokes and/or mouse strokes [Linxi, Habermann 88]. History serves users by allowing reuse of previous interaction.

4.1.2 Artifact History serves Collaboration

Wolf and Rhyne [1992] argue that the process by which information is created and used can be important for understanding of the end product of a work group. In a study done to gain insight into how to facilitate information retrieval in computer-mediated design sessions, they analyzed how group participants used videotape to access meeting information. They found that people searched for information using four main access methods:

- by participant: they remembered person X doing some action
- by communication medium: people recalled what medium was used (eg. whiteboard, overhead transparency)
- by time: people used relative time ("midway through the meeting"), duration ("25 minutes into the discussion"), and clock time ("we only got through item 1.2 by 5 o'clock")
- by relation to other events: people used events as markers before/after other events.

These findings of how people use videotape for information retrieval serve as challenges for computational history mechanisms.

Hutchins' [1990] study of team navigation of large ships also motivates history for collaborative artifacts:

> The work a chart does is performed on its surface— all at the device interface, as it were— but watching someone work with a chart is much more revealing of what is done to perform the task than watching someone work with a calculator or a computer [Hutchins 90p. 217]

The bearing this has to the current work on artifact history is that asynchronously communicating designers do not have the possibility of "watching someone work with a chart." However, by keeping the artifact history, the interaction is available for *watching* at a later time. To relate it to Hutchin's study, imagine a chart which could replay the interaction that took place and show the instruments as they were used.

Design history also provides an approach to design rationale. Though design rationale appears to have

great promise [Kunz, Rittel 70], there have been few recorded successes [Yakemovic, Conklin 90]. The designers must perceive a benefit for the extra cost of documenting their reasoning [Reeves, Shipman 92b]. History is therefore a potential candidate for an interaction tool, because there is no extra cognitive cost associated with having history support. Yet it provides the benefit of restoring the context of previous work, others' as well as one's own.

The benefit of history related to design rationale is that in domains such as network design, which involve two-dimensional sketches and graphical representations, designers can often deduce rationale by seeing the process of how something came to be [Kuffner, Ullman 91; ChenDietterichUllman 91]. A logical map of the current network hides many tradeoffs and compromises that were made in the past, yet which still affect current decisions. Having the history of the evolution clarifies some of the tacit knowledge that is represented in the static logical map.

One side benefit of some groupware aids is that they also help the individual. For example, one designer using INDY said, "What did I do last?" Though the history was primarily viewed as a tool to help one understand other designers' work, it is also useful for reminding oneself of one's own work (sometimes called ''reflexive CSCW'' [Thimbleby, Anderson, Witten 90]). Usually adding multiuser features complicates the system for single users, but history is an example of both kinds of use.

4.1.3 Context is important in Reminding

Research in human memory has shown that people are prone to recall by inferring "what is plausible given what they can remember" [Anderson 85]. Memory performance improves the more closely the current context matches the past physical, emotional and internal context. Much of recall involves plausible inference rather than exact recall [Reder 82].

History tools are needed to support collaborative design. The motivation for this argument lies in the work done in situated cognition relating to context [Lave 88; Carraher, Carraher, Schliemann 85]. Design environments can capture only a portion of the whole context, namely the dates when a given user made certain changes. Yet this small portion can be important in collaborative design.

Each designer on a project team understands only a portion of the overall design artifact. As large projects evolve over time and turnover and attrition take their toll, it will become increasingly important for computer based design environments to help capture the evolution of an artifact and not just its current state. The history serves to remind designers of how the artifact came to be and what the context was when certain decisions were made.

The challenge for computer-based environments is to make that context available which is most useful to reminding. Since there is such a variety of aspects by which people index things, this is a difficult task. For example *circumstantial indexing* describes how people associate seemingly unrelated events in order to aid recall [Bolt 84]. The promise of *embedded history* is to include the dimension of time in the indexing possibilities provided by a computer-based environment.

4.1.4 Embedding History in Artifacts

Researchers have looked at the physics of real-world objects to inform the design of computational media [Hill et al. 92]. For example, as auto parts manuals become worn, they provide visual and tactile cues to guide further use. In the same way that physical wear and tear can be a resource, computational media should embed the history of an artifact in the artifact so that it can serve as guide to further use. This history should be embedded in the design artifact in such a way that the system can leverage the contents of both the artifact and the history. Cross reference from history event to location of the object in the artifact (and back) should be easy.

There are several reasons for integrating history into the design environment rather than attaching it later as a domain independent component.

First, a capability that is deemed useful by designers is simulation, visualizing the performance of a given group of design objects. In order for the history to serve simulation, it must contain a log of design changes as well as past technical data of the graphical building blocks (palette objects). This is especially

important in domains which change rapidly. Network design is a domain in which the basic building blocks are changing at a remarkable rate. An example of this is bridges and routers. The tradeoff used to be that bridges were fast and routers were more flexible. But this distinction is blurred since some bridges now have routing capabilities. By keeping a history of the design environment building blocks as well as a history of the evolution of individual artifacts designed therein, the system can aid the designer in understanding previous configurations. Since the technology changes so rapidly it is likely that one could not make sense of a previous configuration until one saw the technological constraints in force at the time. These are revealed only by exposing constraints or running simulations based on the *historic* capabilities of the design units involved.

Second, in order for a history component to unplay/replay arbitrary sequences of changes, it must have access to the underlying representations in the system. For example, an *edit* function requires before- and after-images to be saved of the edited item. A domain independent component cannot do this. Text-editors appear to be an exception, in that they need to know nothing about the contents of a document, yet can still provide undo/redo. Reasoning about the contents of documents shows, however, that what we would like is not so much undo/redo at the character level, as useful as that might be. As far as a document history goes, what we really want is a history of the conceptual changes. Character undo/redo cannot provide this because the underlying representation (strings of characters) is too impoverished.

Lastly, the history mechanism must support queries that are specific to a given domain. For example, as described in more detail in Chapter 7, network designers requested several kinds of functionality that ranged from domain-independent to domain-dependent:

- "Find where user *x* did something"
- "Trace this Object's history"
- "Undo to the place where I last did something"

These requests illustrate the varying amount of reliance on knowledge of the underlying representations in the computer system. A simple command line history with keyword matching could address the first question about a given user. The second question requires a history for each design unit in the artifact. The third question requires the system to know how to undo design moves. Undoing a *connect* operation requires not only disassociating the item from a group, but also informing the items in that group that they are not longer connected to the given item. A command-line history is not enough to provide this facility.

4.1.5 History and the Language of Designing

As described in Chapter 3, Schoen calls talking and drawing part of the ''language of designing'' [Schoen 87]. The reason that command-line history alone is of little use is due to the impoverished connection between the ''language of doing'' in design and the ''language of the operating system'' [Schoen 87]. The former includes things such as the tradition of the task domain, the environment and media in which design takes place and the means of expression. The latter is an arbitrary language for dealing with computer system abstractions which have possibly little direct connection with the design task at hand.

The language of design should be closely mirrored in the history. Observation of local area network designers in planning sessions, using design tools and in interviews raised several issues related to history tools. The reasoning here is similar to Fischer Lemke [1988b], where it is argued that what users most want is in an important sense not better human-computer interaction, but rather problem-domain communication. In the same way that the computer becomes transparent and allows direct access to problem domain abstractions, the history also must be presented in terms of the user's problem domain.

Designers need a good mapping between the tasks they want to accomplish and the interface language of the system. Well known labels for this mapping are ''gulf of execution'' and ''gulf of evaluation'' [Norman, Draper 86]. Bridging these gaps necessitates representing system capabilities in terms of users' task. For example, during evaluations of INDY, one network designer asked, ''I know I can delete this and create a new one, but how do I *change* it?'' A cursory analysis of this request is that it is domain-independent, since many design disciplines with 2d representations would include replacing graphical objects in their tasks. Yet a closer look reveals that *changing* appears to be a special word in the network design domain: designers and maintainers of the networks do not think in terms of ''delet-

ing'' and ''adding'' hardware, but changing components to add functionality. Rarely are components ''deleted''; rather they are reused in other parts of the network. The question was not how to carry out an edit of a graphical representation, but how to indicate that one device had been changed to another.

A history mechanism surfaces this issue of the ''language of design'', because the history must serve the user in understanding previous work. The language used to describe actions should match closely the language used to do design. If one were to search the history for "change" one would not see "delete" followed by "create". The more closely aligned the history transaction language is with the ''language of doing'', the better the system will be at providing services to augment design. For example, one use of a history would be as a case-based reasoning tool. As the designer proceeds, computational agents search the history for previous configurations that are similar to the current one. These could serve as a catalog of previous designs, to make the designer aware of other possibilities [Nakakoji, Fischer 90].

4.2 Challenges

The goal of a history tool embedded in a design environment is to help designers:

- remember past decisions and so improve the quality of current decisions,
- understand the current state by restoring previous state and seeing how today's situation evolved, and
- communicate and collaborate with others by leaving a trace of one's work.

Table 4-1: Embedded History in terms of Here and Now

Context	Motivation	Benefits
Here	history provides cues for current work	associate history with item that elicits enquiry
Now	need automatic archiving	unrestrained exploration

Table 4-1 summarizes the features of *embedded communication* in terms of *here* and *now*. Since the specifics of the *language of designing* varies from application to application, one will gain the most computational leverage from an application specific history language. Users will come to see the history as part of the artifact rather than as a separate component.

Though *user* interaction histories have proven useful [Lee 92a], a system that supports collaborative design should take the *artifact* perspective. This means that the history is in terms of all changes made to the artifact, rather than several histories stored by user. This provides continuity of design projects as people come and go and the project evolves. The shared artifacts as well as the interaction is kept together to support collaborative design.

Though repetition certainly occurs in user interactions, the real promise for history tools is hardly in allowing the user to page through command lines. Rather, the potential of history is when it is embedded in a computer based design artifact. Then it can support the designer's situated cognition by showing the evolution of a complex design artifact.

Suchman [1987] argues that human computer interaction is constrained by the tendency of the human's knowledge to be context-specific or *situated*. In long-term design projects designers must collaborate across space and time. This means that knowledgeable domain workers will need tools to help remember the relevant issues and see the progress of the various projects.

In a computer based design environment, history tools can provide the context of previous work, rather than just its current state. Understanding ''how'' something came to be often answers many ''why'' questions, especially in design tasks that involve graphics, such as CAD [Sukaviriya 90]. Local and Wide Area Network Design is an area in which logical and physical diagrams play a central role in communicating crucial information. Static diagrams hide much information relevant to modifying the current network, such as how certain tradeoffs and compromises came to be.

4.3 Limits of Embedded History

4.3.1 Social Limits

As evidenced by the special panel on privacy at the CSCW'92 conference, privacy issues have become more and more important as groupware systems have access to more personal data of their users. Though the argument here is that embedding the history of an object with the artifact itself is a good thing to do, there is nevertheless the recognition that saving user interaction history exposes a user to potential violations of privacy.

Since design is an iterative process, one could easily misinterpret intermediate steps as "mistakes." Rather than using the history of the evolving artifact as resources for insight and understanding, one could try to find errors that were made and seek out the people responsible for those errors.

One way to address this potential exposure is to make all history anonymous. No real user names are stored in the artifact history. But collaborators do know each other and benefit from that knowledge, by being able to leave more information tacit. Awareness of the shared knowledge between designers serves them by decreasing the amount of information that must be made explicit.

4.3.2 Technical Limits

The primary technical limit is the issue of level of detail included in the history. For example some text editors save every keystroke of a session, but this is hardly feasible for a long-term design project. INDY confronts this problem by making the history mirror the interaction language. Design moves are kept such as *create*, *move*, *change*.

4.4 Summary

The goals of *embedded history* are to help users remember, understand, and communicate and so improve the process of collaborative design. The motivation for *embedded history* is drawn from an analysis of collaborative design and human computer interaction. History has the potential of serving the individual as well as the team. It can facilitate understanding of previous designs by showing how they evolved. But so far history tools have been studied at too low a level. Tools at the level of command line recall are certainly useful, but there is potential in providing history tools at higher levels of problem solving. The research direction taken here indirectly motivates strong emphasis on task analysis, because the history is best represented in terms of tasks the user wants to accomplish (and later remember).

Situated cognition research on the importance of context motivates history tools that support the collaborative design of complex artifacts [Lave 88; Carraher, Carraher, Schliemann 85]. They must be task specific and integrated with the design environment. The payoff is that the system can then augment human designers by providing important aspects of the context of previous work to serve the task of current design.

5. Empirical Work

This chapter describes empirical work which served the design of INDY. The *McGuckin study* showed the role of critiquing and the co-evolution of problems and solutions. The *Bridge study* raised issues relating to embedded annotations and design history. The *Network Design video* showed the role of deixis and sketching. Finally, the JANUS-NOTES systems provided the first user feedback about textual annotations.

5.1 The McGuckin Study

In light of the goal to provide computational support for design, empirical research was done to explore cognitive issues related to human-human cooperative problem solving. In previous analyses certain issues had surfaced and a large hardware store provided an opportunity to re-evaluate these issues and understand them better. The store served as a success model in addressing these issues and therefore provided insight into how computer based systems can support problem solving and design.

Computer systems should be both *usable* and *useful* [Fischer 87]. For a system to be useful for a broad class of different tasks, it must offer broad functionality. Computing systems have been moving more and more toward high-functionality systems. The more powerful systems become, the more difficult they are to use. Before users will be able to take advantage of the power of high-functionality computer systems, the cognitive costs of mastering them must be reduced. The following problems of high-functionality systems (as identified by Draper [Draper 84], Fischer [Fischer 87], and Lemke [Lemke 89]) must be overcome:

- *Users do not know about the existence of information.* Users cannot develop complete mental models of high-functionality systems. Without complete models, users are sometimes unaware of the existence of information. A passive help system is of no assistance in these situations. Active systems and browsing tools let users explore a system, and critics [Fischer et al. 91a] point out useful information.
- *Users do not know how to access information.* Knowing that something exists does not necessarily imply that users know how to find it.
- *Users do not know when to use information.* In many cases, users lack the *applicability conditions* for information or components. Features of a computer system may have a sensible design rationale from the viewpoint of system engineers, but this rationale is frequently beyond the grasp of users, even those who are familiar with the basic functions of the system. Systems seem imponderable because users have to search through a large list of options and do not know how to choose among them.
- *Users cannot combine, adapt, and modify information according to their specific needs.* Even after having overcome all of the previous problems (i.e., a tool was found, its functioning was understood, etc.), in many cases the tool does not do exactly what the user wants. This problem requires system support to carry out modification at an operational level with which the user is familiar.

One major issue that is not directly related to high-functionality systems but nevertheless plays an important part in their effectiveness is that users do not have well-formed goals and plans. Problem-solving in ill-defined domains can be characterized by the fact that no precise goals and specifications can be articulated. Users of high-functionality systems suffer from a lack of knowledge of the interdependencies between problem specification and which tools exist to solve these problems. Unfamiliarity with this mapping leads users to concentrate too quickly on implementation issues, and they often overlook alternative solutions.

5.1.1 McGuckin Hardware as High Functionality System

To find ways to overcome these problems, we engaged in a search for success models of such systems. Previous research of success models analyzed skiing [Burton, Brown, Fischer 84] and derived architectural components for computer-based learning environments [Wenger 87] from this analysis. In a similar fashion, the ideas behind spreadsheets were used as guiding principles in system-building efforts in other

domains [Fischer, Rathke 88; Lai, Malone 88; Wilde, Lewis 90]. Studying success models can provide us with insights in a similar way that the study of failures [Petroski 85] and their impact on the advancement of design does.

A preliminary analysis indicated that McGuckin Hardware in Boulder, Colorado, might be an ideal candidate for a success model. McGuckin carries more than 350,000 different line items in 33,000 square feet of retail space. The store's superior reputation among its customers and its continued growth and profitability make it a success model.

To get a better understanding of just how the ''system'' operates, we asked McGuckin Hardware for permission to observe and record interactions between customers and sales agents. Some of the dialogs were transcribed from audiotapes and carefully analyzed. Videotapes would have been a superior medium, but would have interfered too much with store operations.

The decision to observe directly as people do problem solving and design in the real world was made as a result of considering the perspective of situated cognition research. Lave [Lave 88], Schoen [Schoen 83], and others have shown how problem solving in daily activity is shaped by the dynamic encounter between the culturally endowed mind and its total context. This leads to a vision of cognition as a dialectic between persons acting and the settings in which their activity is constituted. Lave [Lave 88] argued that theoretically charged, unexamined, normative models of thinking lose their descriptive and predictive power when research is moved to everyday settings and relaxes its grip on the structuring of activities.

The following dialogs illustrate the inherent difficulties in high-functionality systems mentioned above.

Users do not know about the existence of information. In this dialog, the customer is unsure about how to attach a sign to a metal pole. The customer does not know of self-tapping bolts and therefore cannot ask for them. Even if we assume a complete understanding of the problem, this is not enough to guarantee the knowledge of the best tool for the problem. Here the customer ends up buying a fastener that is introduced and explained by the salesperson.

Dialog$_1$: Attaching a Sign to a Square Metal Pole[2]

1. C: *I'm looking for a small fastener maybe one-sixteenth.*
2. S: Okay. Plastic? Metal?
3. C: *Well, what I've got is to fasten a sign on to a square pole. I've got a hole in the top and it fits fine and I got to get one on the bottom.*

   `After looking at several fasteners, and asking a few more questions, the salesperson suggests a certain type of fastener.`
4. S: How about a self-tapping bolt?

   `Picks one up and shows it.`
5. C: *Well, what uhh, well, this would probably do it, what about, would it come back out?*
6. S: Oh sure. It'd come back out.
7. C: *But once it's in?*
8. S: As long as the hole is smaller than this thing, you can thread it in and out.

Users don't know how to access information. The next dialog shows that it can be difficult just to find items you ''know'' exist. The customer is specific about the wanted item and even seems to know the store fairly well, but still cannot find the item.

---

[2]In this and the following dialogs, *C: means customer* and S: means sales agent. `Our explanations and comments are in this typeface.` This dialog is part of a longer one, Dialog$_7$.

Dialog$_2$: Finding Tool Clips

1. C: *I need clips for tools where you shove it up in them and it holds*
2. S: Yeah.
3. C: *I mean not just a single clip, a bunch of them. We tried in housewares, the cheap little ones, tools only have like funny kind of ones. Where else could they be?*
4. S: Garden center, for rakes and shovels and things like that
5. C: *Would it be there?*
6. S: Yeah.
7. C: *Okay, I know where that is, thanks.*

Users do not know when to use information. The interaction shown in Dialog$_3$ involves a search for scales to weigh small animals and illustrates the concept of *applicability conditions*: the conditions under which an item can be used, especially for ''unintended'' purposes. The salesperson is able to recognize a crucial element: namely, that there be a platform large enough to hold something of a certain approximate size and weight. He helps the customer to know *when* to use a given tool, even though that use might not have been intended by the designer of the tool. The fact that a scale is intended for food is less important than those features.

Dialog$_3$ also illustrates the use of *differential descriptions*. The customer describes the intended item ''differentially'' in terms of an example, building on what the environment has to offer. The customer uses an example item (the "little tiny ones over here") to differentially describe the intended solution. The salesperson extracts the crucial information and suggests an item intended for a different domain, yet useful for accomplishing the described task.

In Dialog$_4$ the customer wants strength, but the salesperson points out a crucial feature of that strength: that it comes at the expense of brittleness.

Dialog$_3$: Scales for Small Animals

1. C: *I'm looking for some scales and I saw some little tiny ones over here, but I need something that has a large platform on it, to weigh small animals on.*
   `Holds hands about 18 inches apart.`
2. S: I would think something in our housewares department, for weighing food and things like that. Go on down to the last isle on the left.
3. C: *Okay.*

Dialog$_4$: Hardened Bolts

1. C: *So if I were going to hook something would this be the best thing? What I'm going to have is I'm going to drill into the cement and have it sticking out.*
2. S: You going to have this sticking out, just the shaft of the bolt? `holding a bolt and pointing to the unthreaded shaft.`
3. C: *Right.*
4. S: Hmm. Interesting problem.
5. C: *A hardened bolt would give me more...*
6. S: Yeah, but it'll shear, they're more brittle. I don't know if you'd be any better off with a hardened.

Users cannot combine or adapt information for special uses. Although the combination in $Dialog_5$ is simple, it does illustrate how tools can be combined in various ways. The customer doesn't know why the salesperson suggests a certain combination of tools, but ventures a guess. The salesperson allows the suggestion, but then states his reason.

$Dialog_5$: Combining Simple Tools

1. S: After deciding that a three-sixteenth inch wire is to be looped around a half-inch bolt, which is mounted in cement
   You want a small enough loop, put it between two washers. Picks up two washers and places them on the shaft of a bolt.
2. C: *Small enough loop.*
3. S: Yeah.
4. C: *Why between two washers, so it won't rub?*
5. S: Yeah, so it won't slide off. Probably won't.

Observing interactions like these confirmed the previous analysis of the difficulties of using high-functionality systems. In addition, it raised several other issues that must be considered in building cooperative problem-solving systems.

Incremental problem specifications. $Dialog_6$ shows that there is a close relationship between defining specifications incrementally, as seen here, and establishing shared knowledge, as will be seen in the next dialog. The distinction is a subtle, yet useful one. Shared knowledge has more to do with establishing a common reference point with which to discuss a situation, and less to do with the specific process of identifying relevant parts of the problem domain.

$Dialog_6$: Incrementally Refining a Query

1. C: *I need a cover for a barbecue.*
2. S: (Leading customer down an isle where several grills are lined up and accessories are displayed) Okay... what have we got here... chaise, chair barbecue grill cover... Does that look kind of like what you got? (Pointing to one of the grills.) Similar? No?
3. C: *No.*
4. S: Take any measurements?
5. C: *No.*
6. S: That's a good guess there. (Pointing to a one-burner grill.)
7. C: *It's a double burner one.*
8. S: 52 inches. That's the total length it'll cover. (Measuring with the tape and holding the tape over one of the grills.)
9. C: *Yeah. I know it's not that big at all.*
10. S: You saying about 18 by 18. Well, this is 27, it'll cover up to here. (Using measuring tape again and pointing.)
11. C: *I need two.*
12. S: A couple... in that brand, that's all I have. Here are these Weber ones, thicker material and all that. Here are some smaller ones.
13. C: *I'll take this one.*
14. S: We'll be getting more of these pretty soon.
15. C: *You'll have them by Christmas?*
16. S: Hopefully Thursday.

Achieving shared understanding. Between the time a customer begins to interact with a sales agent and the time the customer leaves with a "satisficing" solution [Simon 81], a shared understanding must be created between the two cooperating agents. The customer must begin to appreciate relevant parts of the

solution domain and the sales agent must understand the problem in enough depth to make reasonable suggestions. Dialog$_7$ shows how establishing shared understanding is a gradual process in which each person participates, sometimes ignoring questions, sometimes volunteering information, and sometimes identifying miscommunications. Illustrated also are the problems of knowing about the existence of information and understanding the results that they produce. The customer wants to fasten a sign to a square metal pole. The top of the sign has been fastened via a preexisting hole, but the bottom is still unattached. The customer learns about certain fasteners while the salesperson learns about the specific problem. Their shared understanding increases as each in turn asks questions and makes suggestions that are critiqued by the other.

Dialog$_7$: Attaching a Sign to a Square Metal Pole[3]

1. C: *I'm looking for a small fastener. Maybe one-sixteenth.*
2. S: Okay. Plastic? Metal?
3. C: *Well, what I've got is to fasten a sign onto a square pole. I've got a hole in the top and it fits fine and I've got to get one on the bottom.*
4. S: Pole have holes in it?
5. C: *Yeah. I had a one-eighth bolt, but it's too big. Need something smaller than that.*
6. S: Round pole? Square Pole?
7. C: *Square pole.*
8. S: (Picking up a fastener and showing it) You tried these?
9. C: (Scrutinizing the fastener.) *Hmmmm.*
10. S: You've got to have a five-sixteenths hole and you fold this thing up and stick it in. Would that work?
11. C: *It's got to be five-sixteenths?*
12. S: Yes. The size of the shaft on this thing. (Pointing to the fastener.)
13. C: *It's not that big.*
14. S: No way to drill it?
15. C: *No.*
16. S: No. What did you use the first time?
17. C: *I tried a one-eighth inch.*
18. S: How thick is the metal in the pole?
19. C: *Oh, probably about one eighth inch.* (Pointing to a certain fastener.) *How about these?*
20. S: (Picking one up and showing the moving parts.) These work on hollow-core doors.
21. C: *Yeah.*
22. S: (Walking over to a different kind of fastener and picking it up) I don't know if this would be strong enough. Still need a three sixteenth hole. If the wind is blowing hard it might give way. Just putting it in with a screw-driver?
23. C: *Yeah.*
24. S: How about a self-tapping bolt? (Picks one up and shows it.) Put that in, tighten it down, (points to tip), that's a thread cutting thread there.
25. C: *Well, what, uhmm, well, this would probably do it. What about, would it come back out?*
26. S: Oh sure. It would come back out.
27. C: *But once it's in?*
28. S: As long as the hole is smaller than this thing, you can thread it in and out.

Integration between problem setting and problem solving. Dialog$_8$ shows an interaction in which a customer wanted to buy heaters, then decided to reconceptualize the problem from one of "adding heat," to one of "retaining heat." This appears to be a trivial reframing and hardly worth notice, but we will argue that understanding exactly this kind of reframing is crucial to building cooperative problem-solving systems. The problem *itself* was redefined.

[3]The beginning of this dialog was also shown in Dialog$_1$

Dialog$_8$: Generating Versus Containing Heat

1. C: *I want to get a couple of heaters for a downstairs hallway*
2. S: What are you doing? What are you trying to heat?
3. C: *I'm trying to heat a downstairs hallway.*
4. S: How high are the ceilings?
5. C: *Normal, about eight feet.*
6. S: Okay, how about these here?
   They proceed to agree on two heaters.
7. C: *Well, the reason it gets so cold is that there's a staircase at the end of the hallway*
8. S: Where do the stairs lead?
9. C: *They go up to a landing with a cathedral ceiling.*
10. S: Ok, maybe you can just put a door across the stairs, or put a ceiling fan up to blow the hot air back down.

5.1.2 Summary

The findings of the McGuckin study related to collaborative work can be summarized as follows:

Simultaneous Exploration of Problem and Solution Spaces. Customers and sales agents worked within both problem and solution spaces simultaneously, or at least alternatively. Typically the problem owner (customer) had a better grasp of the problem space and the problem solver (sales agent) had a better understanding of the solution space, and over time these spaces converged until there was a large enough intersection of shared knowledge within which potential solutions could be evaluated. This is seen in Dialog$_7$ in which the customer knows what needs to be done but needs a better understanding of the possible solutions, and the salesperson knows how many different fasteners work but needs to understand the specific application.

This finding influenced the design of INDY in two ways. First, there is computational support for designers to challenge each others' view of the problem at all levels. Rather than forcing annotations to be only at the level of specific design units, there are no restrictions on where annotations can be placed and no structure in terms of categorization is enforced. Second, since the understanding of a design task changes over time, designers should have the opportunity to watch this process. This motivates a design history which captures more than just states of an artifact at predefined intervals, but rather captures the design moves as they occur.

Critiquing shapes the design process. The interaction between customers who were solving problems and salespeople who assisted them could often be described as one of critiquing. Both participants made suggestions, corrected the other, and asked for help in brainstorming. During the problem-solving episodes, the main problem was rarely fully fixed, but was still open to questioning; both participants continued to narrow the focus, but also to challenge the focus.

This finding informed the design of INDY as follows. Since critiquing plays such a basic role in collaborative problem solving, INDY supports human-human critiquing. Sometimes customers would make a suggestion verbally and other times they would carry out the suggestion directly using the materials at hand. INDY supports both kinds of critiquing, making textual annotations as suggestions, or altering the artifact itself to implement the suggestion.

5.2 The Bridge Game

Habraken and Gross [1988] used design games to focus attention on a single aspect of design. To explore issues of human-human critiquing related to design artifacts, a pilot study of critiquing was carried out: the *bridge game.* This bridge game was done as part of a graduate course and involved pairs of cooperating subjects building and critiquing bridges built with Lego^{tm} blocks.

Since the focus of this thesis is on the ''different time/different place'' (DTDP) quadrant of Figure 2-1, the bridge game was implemented as an asynchronous process: one person designed, then the partner critiqued, then the designer redesigned. Technically this was ''different time/same place'', but the nature of the game was such that the issues exposed were also true of the DTDP quadrant.

Critiquing was limited to textual notes and no spoken communication was allowed. This was to simulate the DTDP environment in which direct communication (face-to-face or telephone) is not possible. As argued above in Chapter 2, design views include drawing, constructing and arguing. To analyze in more detail the role of drawing and constructing, a study was done to explore the following four questions.

1. Is sketching much faster than building with physical blocks? This question was motivated by the pervasive use of sketches by network designers and the views of designing as drawing, constructing and arguing.
2. What kinds of annotations will the person who critiques the sketch use? This is motivated by observations of how network designers overwrite white boards and architects use tracing paper to communicate suggestions.
3. Are critiques best communicated by doing or describing? Some types of changes are easier to do than to describe, but doing something can lead to problems of misunderstanding, the purpose of the change might not be clear, or oversight, in which case the change goes unnoticed.
4. How obvious are design goals from the design artifact? This was a variation on other design games in which the task was for one person to infer the design rules invented by another.

5.2.1 Method

Four pairs of participants were given goals that involved building and critiquing bridges. Three of the four pairs used lego blocks to build the bridges; the fourth pair used pen and paper to sketch. Each of the pairs had slightly different design goals, and within a pair, some partners had different design goals.

Table 5-1: Goals for each pair of participants

Bridge Design Goals		
Group	**Designer**	**Critiquer**
1	One each Scale, Aesthetics, Strength	Guess which one is which
2	Scale	Strength
3	Aesthetics	Aesthetics
4	Strength	Scale

Group 1 was given the task of doing three bridges instead of just one because it was thought that the sketching would go much faster than building.

Group 3 was the only group in which builder and critiquer had the same goal. In group 2 the builder's criteria was scale, but the critiquer's was strength. Group 4's goals were just the opposite. The purpose of the conflicting intra-group goals was to see whether the critiquer would notice that the bridge had been designed according to alternate criteria and conversely whether the designer would notice that the critiquer had ''misunderstood'' the criteria.

The purpose for including a group using sketches rather than lego blocks was to see how the medium of the artifact affected both design and evaluation. In two of the groups, the critiquer used evaluation criteria that differed from that of the builder. Neither critiquer noticed that the bridge had been designed

according to other criteria, but both designers noticed that the critiques were based on criteria that differed from their own.

5.2.2 Results and Discussion

Sketching vs constructing. The sketch group created three bridges and the other groups created one bridge. It is hard to generalize, but it seemed that the sketcher was careful to illustrate the various design goals clearly, whereas in design meetings, sketches can be very rough since speaking and gesturing aids in understanding. Sketching seems to play such a central role in design that it was included here to see how it compared to ''constructing,'' and how it affected critiquing.

Based on just this one example, sketching tools for asynchronous communication need not have quite the usability of pen and paper, since there appears to be much more deliberation and deliberate drawing than in face-to-face meetings in which speech and gestures communicate a bulk of the message.

The use of overlays in critiquing. The sketch critiquer did not make use of the transparencies in a 2d overlay mode. He just used it as a writing surface. This served as a reminder that it is difficult to anticipate users' preferences and use of tools. Sketching is not always better for everyone.

Describing vs implementing critiques. All textual critiques appeared to have been fully understood, so this game did not show that doing (rather than writing) leads to fewer misunderstandings. The rebuilder started over from scratch with a different design and this brings up an interesting issue: when a critiquer says ''change this to that,'' there is no guarantee that it is even possible. If one implements the critique rather than writing about it, then the guarantee is made — it is obviously possible, because it has just been done. The tradeoffs between delivering critiques by describing them or doing them are summarized in Table 5-2.

Design Goals and Artifacts. The critiquers did not guess that the bridges had been designed according to alternate criteria. But the designers did guess that they had been evaluated according to alternate criteria. Writing on the part of the critiquer helped make tacit features explicit.

In real-world design, part of the difficulty in team-work is that members are working under different assumptions or understandings of the goal. In this game, the critiquers did not notice that the bridges had been built according to other design goals. Although one could argue that it would be difficult to notice this given the scale and makeup of the models, it still is worth noting, because the designers *did* notice the discrepancies in the critiques. This was because at first, there were no explicit markings on the bridges. But after the bridges were critiqued in writing on post-it notes, it was relatively easy to guess the critiquer's criteria, e.g. ''How about connecting the upper spans for extra stability?'' Some notes even referred to the criteria explicitly, e.g. ''Nice symmetrical design, but how does this design reflect the goal to support truck traffic?'' Written critiquing makes design goals more explicit. That is, design goals which were implemented tacitly are made more explicit by the act of writing a critique.

Table 5-2: Critiquing Method: Describing vs Doing

	Describing	Doing
Advantage	simple tools: paper & pen; elicits tacit knowledge by forcing certain features to be named and described	not forced to make things explicit; avoids impossible critiques
Disadvantage	can't go ahead and make trivial changes; can lead to impossible suggestions	critiquer must know how to manipulate artifact; could accidentally undo valuable or time-consuming work; changes could be misunderstood; changes could go unnoticed

The critiquer in group 4 was told to simply rebuild the bridge according to certain criteria. This was to explore the relationship between writing and talking about critiques and doing them instead. It pointed out a feature that design environments should have: a history of design artifacts. After the critiquer rebuilt it, the original designer was given the opportunity to comment on the redesigned bridge. Certain

features were hard to argue about, because it was hard to remember exactly what the previous bridge looked like. Later the audience had to ask the group what some of the comments meant and how the bridge had looked before.

The original designer's critiques of the redesign were in terms of the tacit features he had implemented at the beginning. My own experience as project lead confirms that this is a pervasive problem in design discussions: people critique one design from the standpoint of an unavailable or undemonstrable previous design. A design history would be invaluable here, because the arguments could all be grounded in the artifact as it really was, not as different people remember it.

5.2.3 Additional Observations

The study raised two further issues related to computer based support of collaborating designers. First, critiquing is influenced by the artifact. Other studies have shown that there are differences in spoken vs. written evaluations, i.e. that the critiquing medium affects the critique. Since one focus of this research is embedding the communication in the design artifact, this then becomes an issues relative to the critiques. The study done by Chalfonte etal [1991] indicates that there might be differences between embedded voice annotations and embedded textual annotations.

In the bridge study, detailed comments were placed in a way that they were close to the content of the note, i.e. attached to the parts to which they referred. More global remarks were placed on larger notes and placed near the artifact. However, there are exceptions to the sense of appropriateness. Even in this small study, one ''obvious'' use of a medium, the transparencies, was ignored, and instead of being used as an overlay on which one could circle areas and draw arrows pointing to parts of the artifact, the transparency was just used as a writing surface, much like an overhead transparency with bullet items.

In this game, critiquers took advantage of the glue on the post-it notes to place them at appropriate places on the lego bridge. In one case this caused a problem. The note was placed underneath the bridge and the designer did not see it. The physical make-up of the artifact affected how and where notes were placed.

The second issue relates to the tools needed to support asynchronous human-human critiquing. Writing tools are an obvious beginning. Written language is flexible enough to allow just about any critique to be expressed. However language is not always the easiest. Sometimes doing is easier than explaining. In this case, access to the artifact is a necessary tool. The critiquer should be able to carry out tentative design decisions — to explore what-if scenarios before committing to a particular critique. This can prevent ''impossible'' critiques, one in which a suggestion is made which is not possible given the current constraints.

5.2.4 Summary

This small study of critiquing illustrated several issues in asynchronous collaboration.

Notes can obscure and clutter the artifact. At least for the scale of the bridges in this game, notes quickly obscured most of the artifact. The problem with removing them from the artifact was that it was easy to loose track of where they had been. In disciplines in which 3d mockups play a large role, such as architecture, it is clear that this finding does play a role. The scale that seems to be easy for people to generate and evaluate is small enough that any amount of legible notes begin to clutter and interfere visually.

This maps well to users' experience with early versions of JANUS-NOTES. It does not take long for notes to interfere with the artifact under construction. A way to manage notes is crucial so that they do not get in the way and interfere with the designing. The glue on post-it notes can be mapped to connectedness in a computer-based tool. Whereas the glue usually leaves no trace and thus the exact location of the note can be lost, a computer tool can easily allow the ''note'' to be scaled/moved/hidden all without losing the original connection to the artifact. This is one way in which computer tools can transcend the current work habits of annotation.

Notes get lost in the artifact. This is related to the previous point. One post-it note was stuck on the bottom of a bridge and not noticed by the designer. In a computer-based design environment, it is not

enough to provide tools that allow the user to get the notes out of the way, there must also be ways to easily access relevant information. This information could be spread out across several notes, placed over a period of time.

A design history needs to be kept of all changes. This is another case in which computer based tools can transcend common work practice. In this author's experience with large system development (multi-year, multi-person systems) the best one did was to save the state every so often. One would accumulate stacks of documents that someone thought might be relevant in the future. It was not easy to predict what might become relevant in the future, but no one wanted to save everything. This led to major sins of omission. A person would leave the project and somehow his knowledge as well as documents would leave with him.

Computer based tools can transcend this practice. Actions can be captured and stored as design transactions. This takes no effort on the user's part, but provides immediate payback in terms of "unlimited undo." Other tools can be built on top of the history mechanism to provide features such as dynamic catalog exploration, design rationale, temporal context-specific help. These issues will be discussed in more detail later on.

Design goals are not obvious from the artifact. One could take this several ways. One approach would be to force users to make design goals explicit upfront, such as systems that translate specification languages into a finished product. However situated cognition research shows that we will never be able to overcome the fact that plans are merely one among many resources which designers use [Suchman 87]. It will only work with well understood problems.

A better point to take from this observation is that tools should be provided which encourage disclosure of tacit knowledge without the negative impact of forcing too much explicit structure. In the case of INDY, the design history serves this purpose. It takes no extra effort on the part of the designer, yet provides several important services to collaborating designers.

The bridge game, though informal, illustrated several problems with asynchronous human-human critiquing. Insights from the study agreed with experience using INDY and analogies with post-it notes and transparencies affected implementation of the system.

5.3 Network Designers Design Session

Schoen's protocols show that deixis plays a large in architecture design sessions [Schoen 83]. To verify whether this tight integration of talking and doing had more to do with architecture than with design, I observed two network designers planning the installation of new workstations. The transcript revealed that in 660 sentences there were over 120 deictic references.

B: Okay, it's a bridge, okay, good. So we have our thinnet, and off the thinnet hangs the gatorbox. but also off the thinnet hangs sigi.
H: And symbolics.
B: Off the ThinNet?
H: They have those black box.
B: That's right. C3D1 and 2? And where are all the Macs in cr zero twenty?
H: In cr zero twenty they are on the AppleTalk local AppleTalk [here.]
B: [These] are CR020 Macs.
H: But [this] network goes all the way up to the seventh floor.
B: Right. It's all over the place.
H: Snakes all over the place.
A: That's what we're going to talk about.
B: Including Harriet's office and Lenore's office. So we can. Harriet, Lenore what about Clayton's Macs? Where are they? They are on [here] too, right? 118 Macs. Gita is on internet, so Gita is [here].
H: Actually, she has the repeater up [there].
B: Where is Mike?
H: Mike is on [there].

Figure 5-1: Deictic References

References to sketches pervade spoken discussion. Deictic references are [boxed].

These references to the sketches which were made did not make sense apart from the video portion of the design session. The tendency of anchoring discussion with artifacts is not restricted to architecture, but appears to hold true for disciplines in which it makes sense to produce sketches and pictorial representations. Figure 5-1 shows a sample portion of the dialog of the network designers.

This finding suggested that INDY should allow users to make these kinds of references easily, by means of unstructured textual annotations.

5.4 JANUS-NOTES

INDY evolved from an extension to JANUS, a design environment for kitchens, called JANUS-NOTES. It was an addition to JANUS which allowed textual annotations to be placed in the work area. These notes were associated with a design unit, so that one ''annotated'' objects in the work area. One could also annotate notes themselves. This addition provided feedback on certain basic aspects of integrating text annotations and the design artifact into the same medium.

Four members of Human Computer Communication research group at the University of Colorado and one visiting professor used JANUS-NOTES for about 20 minutes each and commented on the interface and functionality. The evaluations were informal and I took notes of the comments as they were spoken. Figure 5-2 shows a simple example of annotating a kitchen.

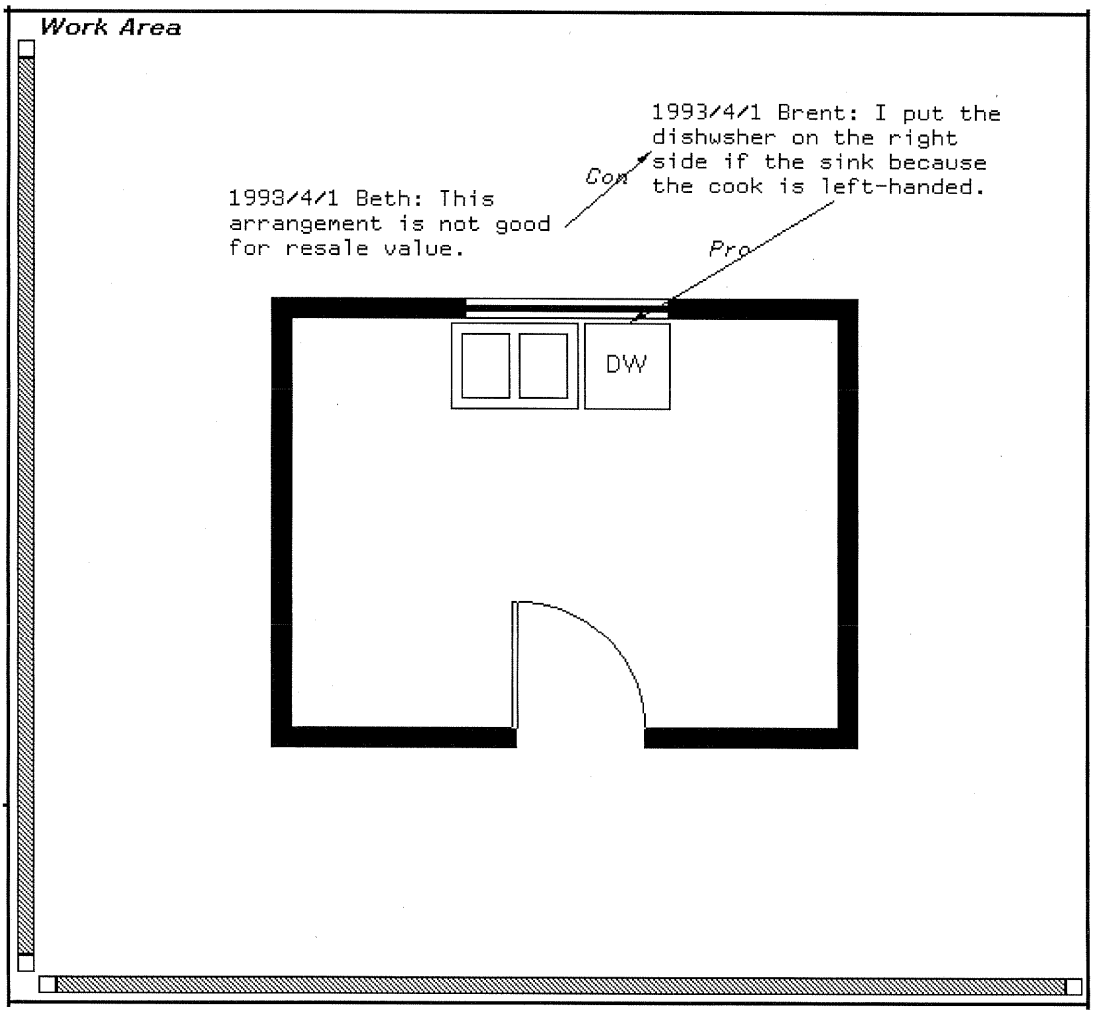


Figure 5-2: JANUS-NOTES

JANUS-NOTES supported context-specific arguments and enforced typed links between annotations.

The most often mentioned comment was that there needed to be a way to manage notes. Verbs that were mentioned in this context were hiding, shrinking, organizing, and suppressing. These verbs guided the design of the command names in INDY.

Managing notes was necessary because the display quickly becoming cluttered after just a few notes. The relative size of design-units and notes, and the minimum font-size for readable notes caused the notes to

quickly take over the major portion of the screen. Design unit icons can be scaled quite small and still be recognizable, but notes quickly loose their semantics when scaled beyond legibility. At even tiny scales, design units can be recognized, but there are not enough tiny fonts to allow one to recognize notes that are scaled very small. One thus looses the overall shape of the note, since beyond a certain scale, ellipses are printed.

Another comment addressed link types. In the version of JANUS-NOTES that was used, notes were always attached or connected to another design unit. One could not just create a note as a stand-alone text box. This forced users to choose a link-type. Though users commented that it would be nice to have two things: 1) a default link type, and 2) a user-customizable link-type, the consensus seemed to be that one should be able to create notes without any need to choose link types.

Though a default link type would allow users to enter text without being forced to be explicit about the relation to other design units, even choosing the default was considered annoying. This confirmed the idea that it is almost always preferable to users to allow them to do as much as possible without forcing upon them the ''tyranny of the explicit'' [Hill 89].

5.5 Summary of Empirical Investigations

Empirical investigations and evaluations of early prototypes guided the design of INDY. The study of people collaborating to solve problems showed:

- Problem solving is a cyclical process in which partial solutions cause a reframing of the original problem. Problems and solutions co-evolve.
- Critiquing plays an important role in collaborative problem solving. The problem framing itself is open to questioning.

The study of people collaborating asynchronously via written notes showed two issues related to notes: 1) they can clutter the design artifact, and 2) they can get lost in the artifact. Two other findings were that a design history is useful for understanding critiques related to the current design, and that design artifacts ''do not speak for themselves'' [Rittel 84].

Observations of network designers showed that sketching and deixis play a central role in collaborative design. Early evaluations of INDY prototypes confirmed the need to manage notes and remove barriers, such as premature structuring, to their use.

6. INDY

INDY is a system which supports the collaborative design of computer networks. It builds on design environments [Fischer 92a] by providing a design medium in which the design as well as the communication about the design is stored. After a description of the computer network domain, a scenario illustrates the main features supported in INDY. Then *embedded communication* and *embedded history* are considered in more detail.

6.1 Network Design Domain

A local area network links devices such as workstations, file servers, and printers using cables of different lengths and types. Networks can be viewed at different levels of abstraction, such as connectors and cables, work groups, and entire subnets. Beyond the subnets are building-wide backbones and eventually extremely large nets such as arpa- and internet.

Though a fair amount of trouble-shooting is involved in maintaining a network, a large part of administration is more open-ended designing. Technology continues to move forward at such a rapid rate that basic assumptions are frequently challenged. As an example of this, consider ''Twisted Pair'': it was considered old technology until recently, when engineers discovered ways of increasing the bandwidth by orders of magnitude. Plans that had been made to phase out twisted pair had to be reconsidered.

The domain has hard and soft rules. For example, CSMA/CD networks must have a hierarchical topology. An example of a soft rule is that RS 232 requires communication lines to be less than 75 feet in length, but experience has shown that in normal circumstances they can safely be extended to several hundred feet. Sources of design knowledge for this domain are books [Nemeth, Snyder, Seebass 89; Tanenbaum 81; Comer 88], standards such as IEEE 802.3, product literature, and local experts.

Rarely are networks designed from the ground up. There is usually too much investment in the old infrastructure and adding functionality is a process of evaluating difficult tradeoffs. It seems to be worse than software in that most of the work is done after the initial design is allegedly over. Just like in a large software project it is crucial to have a chief architect who oversees the project from start to finish, so too does a turnover in systems administration make it difficult to make good decisions about tradeoffs [Brooks 83]. There is rarely one ''right'' way to build a complex network at the level of, say, the Engineering Center. Thus personal experience and bias plays a large role in redesigning existing networks.

Since networks serve so many kinds of processes, the expertise required to understand them all is rarely in one individual. Experts in different areas are required to work together to meet constraints such as cost, efficiency, and expandability. Though technological advances continue to make network design easier, it is still true that mundane things like elevator shafts and access to cable routes have a non-trivial impact on networks. Thus some architecture firms specialize in ''computer network'' buildings.

This domain is useful to study from the perspective of this research, because interpersonal communication plays such a large role. There are many ways to support given requirements and problem framing plays an important part. The technical nature of the field seems to have affected the people's design process less than the human nature of the designers has led to certain design processes.

Network design is typical of a wide range of other design domains. Though analysis tools exist, it is not possible to automate network design. Each solution illustrates tradeoffs between many goals such as cost and reliability.

Design Representations. On their own, designers in various domains of expertise have evolved certain representations to meet the above needs. The domain of interest is that of local-area network design. Over time, the logical map has evolved to serve as a key design document. It abstracts away many physical characteristics of the network but preserves connectedness. Types of devices are represented by icons that were agreed on by consensus of the community of network designers. The approach here to tools and artifacts currently in use by network designers might best be described as ''respect:'' respecting the tradition, yet still seeking ways to transcend everyday work practices [Kling 91].

Figure 6-1 shows an example of a logical map, used to ground discussions about changes to the (real)

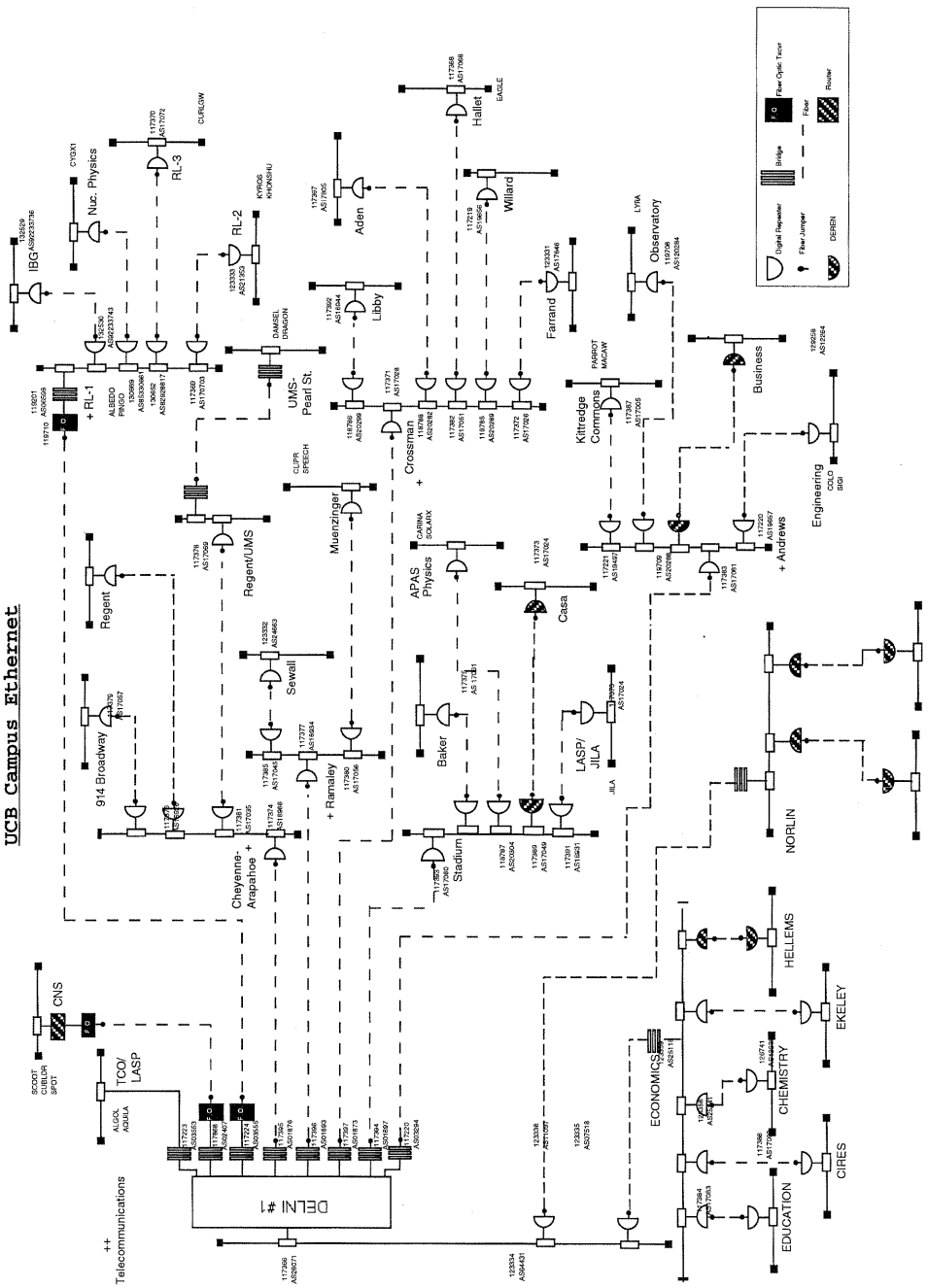


Figure 6-1: A portion of a Logical Map from a Network Designer

network. This map is a portion of a MacDraw^{tm} document created by a network designer. This artifact was used to focus issues that arose during design meetings. One can write on it directly, annotating and arguing. It is easy to do ''end-user modification'': one just adds a symbol in the lower right-hand side box and then does a cut-and-paste onto the document. This representation has the same limitation that any domain-independent tool has: changes to this document have no semantic meaning beyond the interpretation given to them by the users.

Figure 6-2 shows another logical map produced by a computer tool, the MIT LCS Interactive Map program. This tool goes beyond the static document depicted in Figure 6-1 by adding domain-specific support and active components, e.g. an SNMP (simple network management protocol) component that allows the user to select a node and query the network to see its current state. The downside is that the system is not easily changed to accommodate new devices: one cannot just "add a new icon" and start placing it in the network, as in the case with the MacDraw document.

Both types of design artifacts shown in these figures influenced this work on a network design environment. These diagrams are similar to those used extensively by network designers in videotaped sessions to explain the history and rationale behind previous decisions and to discuss possible revisions when given ''what if'' scenarios. The videos illustrated the importance of grounding one's interactions in an appropriate representation of the artifact being designed and interpreted.

6.2 Scenario

Consider two network designers, Joshua and Luke, who are responsible for a network which has grown to the point where traffic flow is congesting the network. The growing number of workstations is placing an increasing demand on the bandwidth of the network which in this case is a single ethernet backbone. The current performance degradation as well as anticipated growth present a problem which is not completely specified. Nothing is actually broken, and it is hard to measure the costs of a congested network. Yet there is enough motivation to cause action. The current configuration is shown in Figure 6-3.

Joshua decides that the best way to alleviate congestion is to partition the network and so isolate traffic. In anticipation of growth, rather than just splitting the backbone in half, a separate cable is laid which for the most part duplicates the current one. There are difficulties in the physical routing of the cable so that it is not a complete parallel. The old and new cables are connected via a bridge so that they are not isolated from each other. In evaluating where exactly to split the new cable, Joshua considers the building layout and places a bridge at the point where one building wing joins the main part of the building. Figure 6-4 shows the results of these changes. In doing this he has used INDY as a construction kit, creating and changing domain-specific design units.

The infrastructure is now in place to segment the backbone and isolate traffic flow. A router is installed which segments the now-two-logically-distinct backbones. An analysis of the backbone cables and the physical locations of users and workstations requires that the router be placed at a different location than the bridge (See Figure 6-5).

A few days later, Luke logs on to check the proposed design. Since he was involved in the original installation of the Symbolics, he knows that the proposed split will cause a problem with the Symbolics workstations and suggests extending the Office Tower cable via a repeater and thinnet to accommodate the need for these to be on the same network. He leaves Joshua a note explaining the changes. He can leave the note right where it is most useful, as seen in Figure 6-6. Joshua had thought they were on the same network because the two cables were connected by a router, but the workstations did not support a protocol at the internet protocol level, but the ethernet level. Luke has used annotations to critique the design and by doing so has surfaced a requirement which had been hidden until now: that these workstations can be separated only by a bridge or a repeater, but not by a router.

Much later, after workstations have undergone changes in networking support, Benjamin is charged with adding another symbolics workstation. He sees that the current Symbolics are attached to thinnet via a repeater and adds the new one along side them. However he notices that the thinnet parallels the thicknet backbone and wonders why the thinnet cable was run in the first place. He happens to know that Symbolics should be on the same network and also that they support all levels of network protocol shown in the logical diagram. Given the current level of software and hardware technology, the present configuration makes no sense.

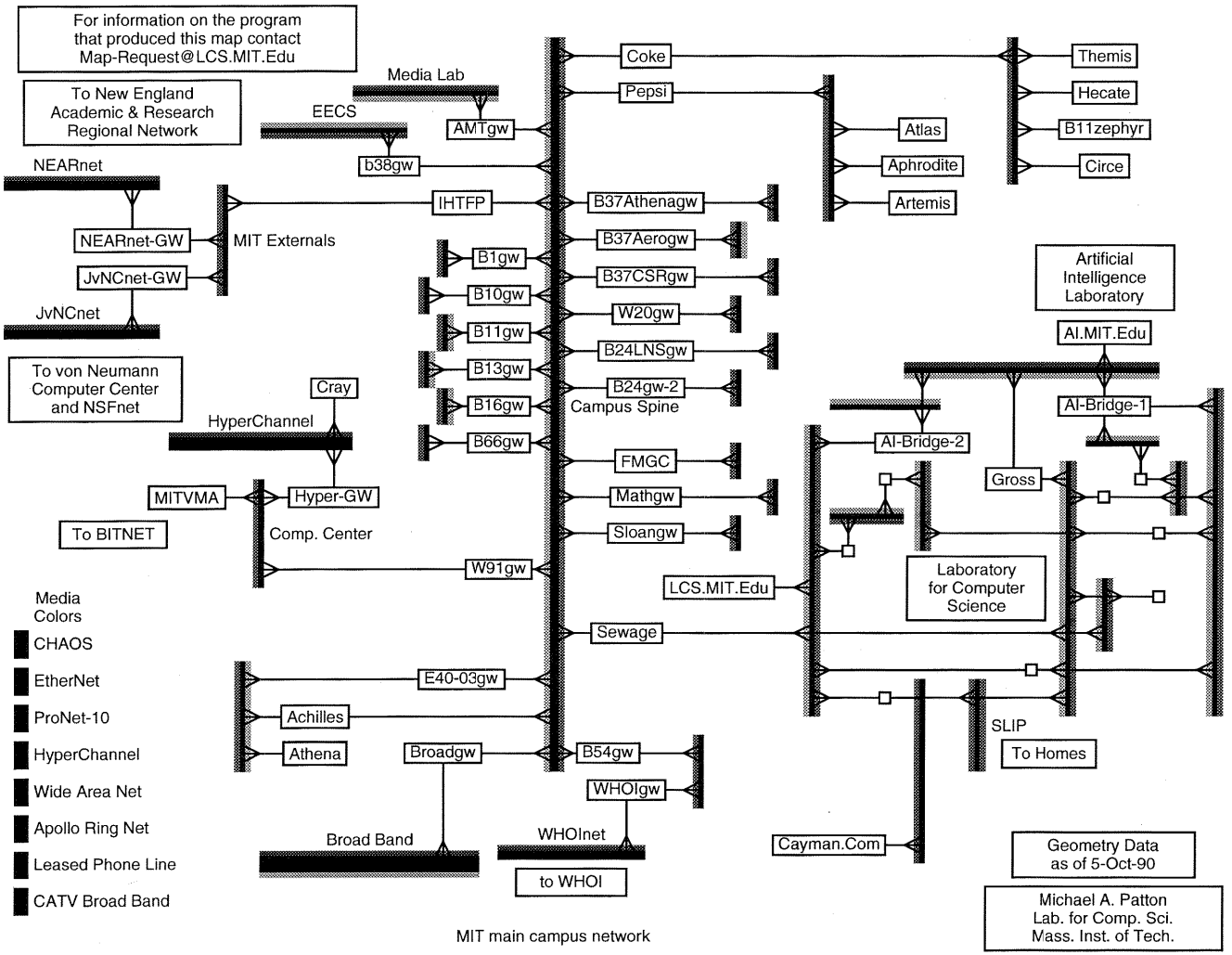


Figure 6-2: Output of Computer Tool used by Network Designers

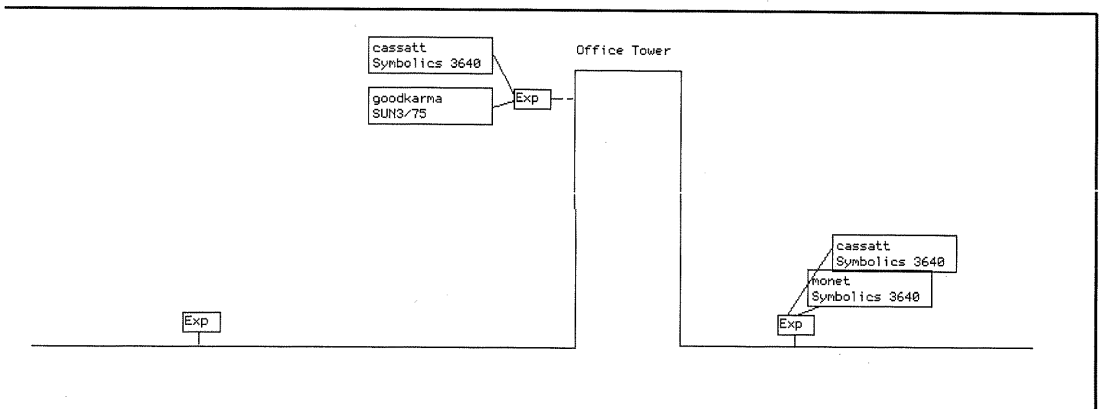


Figure 6-3: Scenario: Beginning Configuration

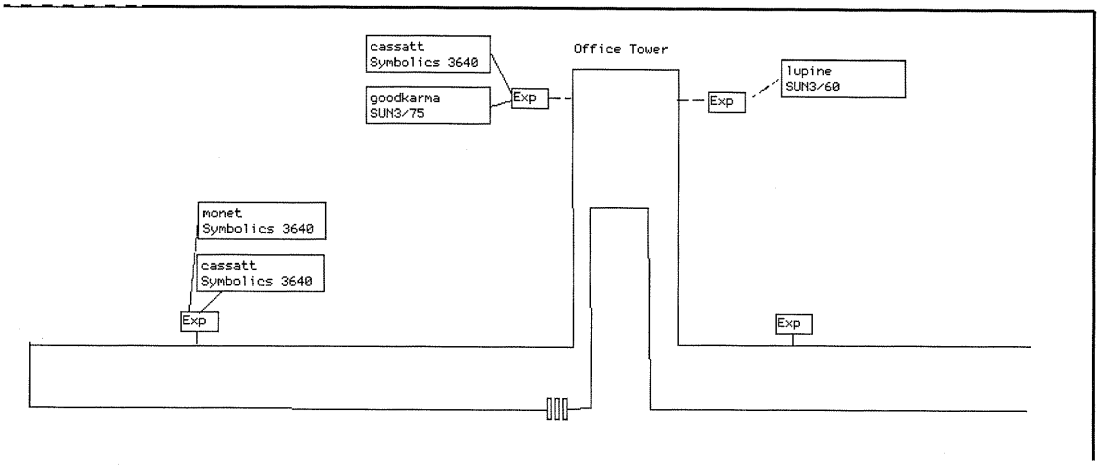


Figure 6-4: Scenario: After installation of Bridge

So Benjamin decides to investigate why the symbolics are setup as they are. He inquires about the history of the repeater and finds it was created on 6/29/92 (Figure 6-7). He rolls back the artifact to that point in time (Figure 6-6). He explores the history related to the backbone split and notices several notes about the symbolics placement. He checks the attributes of one of the symbolics workstations and finds that at that time, they did not support the ip protocol level. Satisfied that he understands how the current situation came about, he returns the artifact to the current date.

Since the current technology supports direct connection to the backbone he changes the new machines from the thinnet, where he had placed them, over to the thicknet backbone and leaves a note saying he's done with the design (Figure 6-8).

Later Joshua is working on another part of the network and notices changes have been made. He replays the steps taken by Benjamin and sees how he had first hooked them up to the thinnet, then changed his mind. He disagrees with the placement and writes a note arguing that even though it is possible to hook them up to the backbone, that will create traffic on more of the network than keeping them on thinnet. He uses sketching tools to group the workstations together and show how they relate to each other and the subnet. This illustrates the view of design as arguing tradeoffs. There is no real best solution, just tradeoffs given the current context (Figure 6-9).

This brief scenario shows design as drawing, constructing, and arguing. In this scenario, the tasks in-

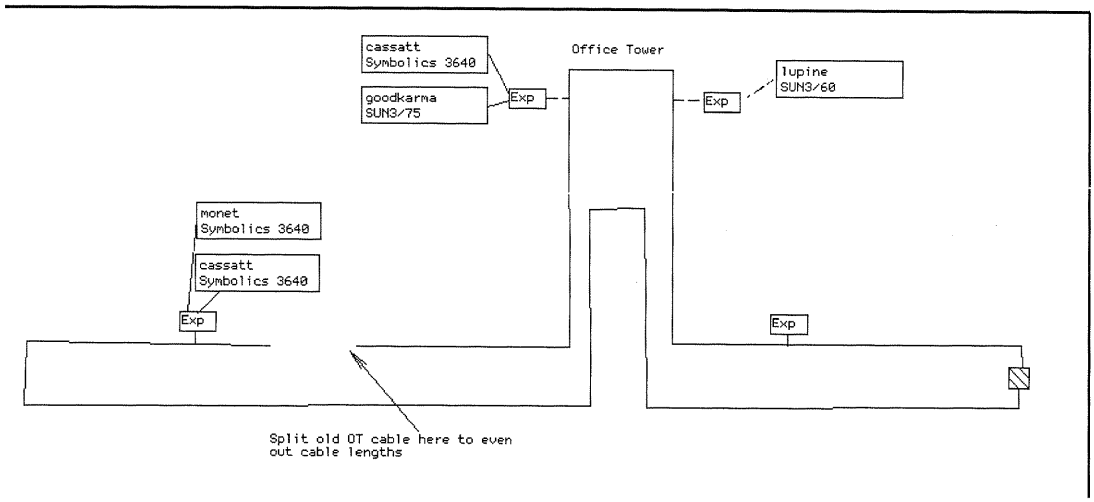


Figure 6-5: Scenario: Backbone is split

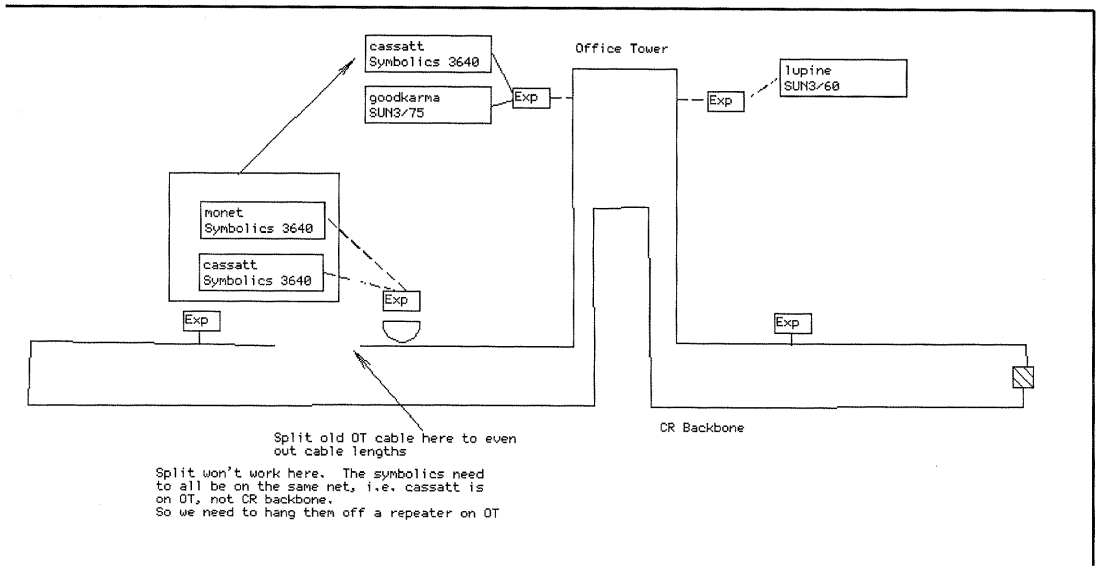


Figure 6-6: Scenario: Critiquing Design Decisions

cluded decreasing traffic flow while maintaining required connectivity and adding new workstations to the network. The situation and task at hand elicit communication and INDY allows that communication to be expressed directly in the context. Furthermore it need not be delayed, but can be done as the task elicits the need to clarify or argue. The history is accessible from any design object, and serves the task at hand by helping a designer understand the current configuration.

The network situation described is very similar to the current configuration in the University of Colorado Engineering Center, and the tentative move of splitting the backbone and separating the workstations would have been carried out had not one of the network designers remembered the constraint related to the Symbolics workstations being on the same subnet.

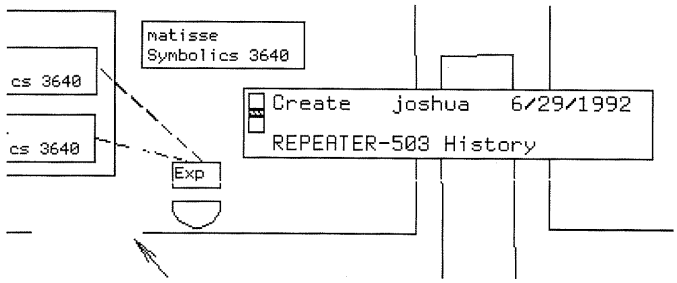


Figure 6-7: Scenario: History Trace of Repeater

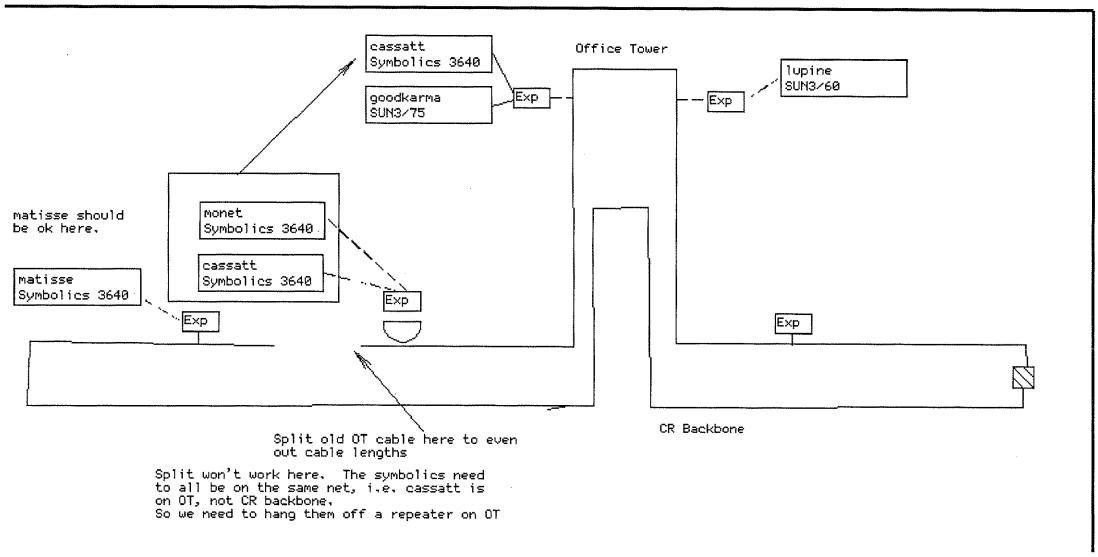


Figure 6-8: Scenario: New Machines can communicate through Routers

6.3 Embedded Communication in INDY

INDY supports asynchronous drawing, constructing and arguing design issues.

Drawing. Design and planning sessions of network designers show frequent use of sketches. INDY thus supports 2-dimensional graphics. After analyzing graphical artifacts produced in design and planning meetings, the following graphics objects were included in INDY: *line*, *arrow*, *rectangle*, and *ellipse*. Figure 6-10 shows the palette of these objects as well as their use in the work area.

Construction. INDY uses the toolkit approach to support domain-specific construction. The palette (shown in Figure 1-2 above) contains domain constructs and these are selected and placed in the work area.

The palette supports several tasks: 1) creating representations of existing design units, 2) adding new design units, and 3) adding new items to the palette.

First, one should be able to create depictions of existing network entities without having to re-enter the data. For example, if the user is adding a subnet to the work-area that already exists in the campus network, then the system should allow the user to identify the object from the network data and place it, without having to reenter attribute values. Thus adding the symbolics network to the work area should be a matter of choosing which machines to add, not typing in data for each machine.

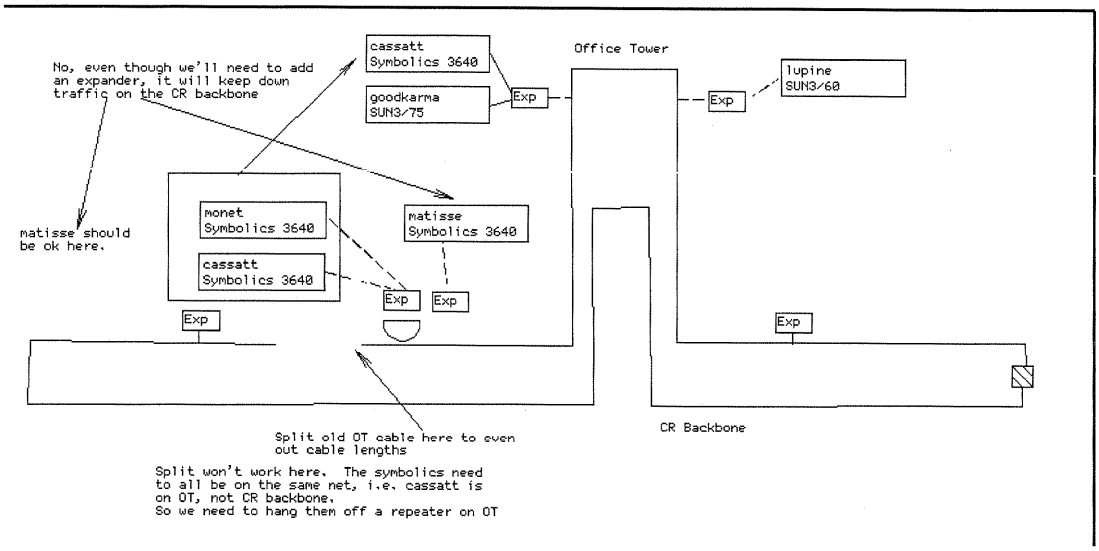


Figure 6-9: Scenario: Final State of Network

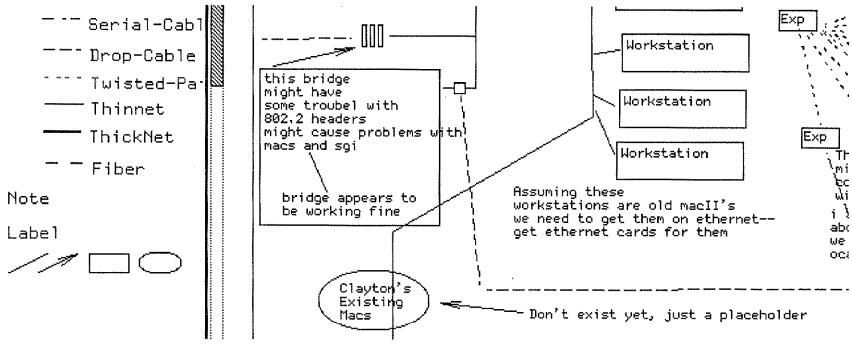


Figure 6-10: INDY Graphics Objects in Palette and Work-Area

The INDY workarea supports simple graphics objects. Lines and arrows indicate connectivity. Rectangles and ellipses show groupings. Here a line connects a clarification of a note, the two notes are grouped in a rectangle and the rectangle points to the bridge. The ellipse clarifies that the note "Clayton's Existing Macs" represents an object in itself and is not a comment on existing design units.

Three ways exist currently to represent existing entities: choosing by name, room, or ip subnet number. If the user knows the name of the machine, he can enter it or choose it from a menu of all machines. If he knows the physical room number, he can enter that, then choose from a menu of all the machines known to be in the room. Or he can choose a machine from a menu of all machines on a certain subnet. When the user chooses a machine, the attributes and values are instantiated in a design-unit in the work area.

Second, one should be able to add new design-units to an artifact to explore alternatives or do simulations. Here the design-unit is instantiated with default values which the user can change for simulation purposes.

Third, one should be able to add items to the palette. This is usually for one of two reasons: 1) A design unit is configured in a way that is frequently reused, such as a certain type hard disk, a certain amount of memory, etc. 2) A subassembly of several design-units is frequently reused. Figure 6-11 shows an

example of a group of components that were suggested as an example of reusing groups of built-up sub-components.

When a design-unit is added to the palette, then later instantiated, it is copied to the work-area just like any other palette item. The user can then modify the unit to make it specific to the task at hand.

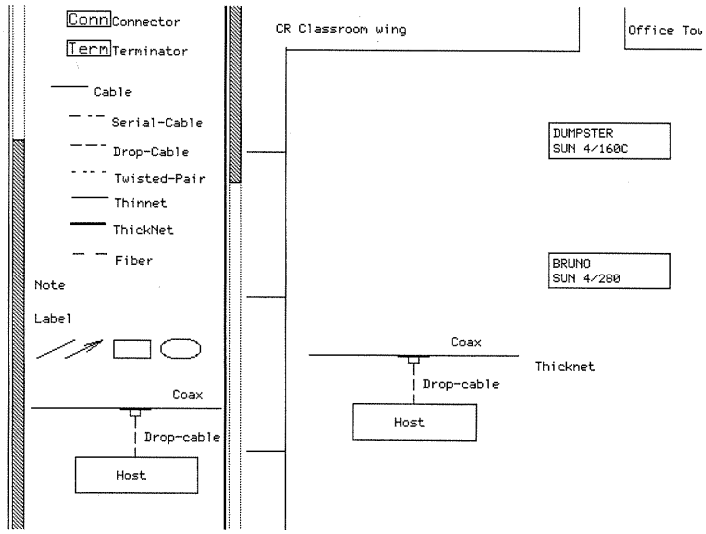


Figure 6-11: Example of Palette additions

The modifiable palette allows users to build sub-components for reuse.

Argumentation. The type of argumentation supported in INDY is situation specific critiques through textual annotations. Though a more general issue base is envisioned, the primary focus in INDY is supporting context specific argumentation that arises during real use of a system (Figure 6-9).

Another way INDY supports argumentation is by *implemented critiques*, which distinguishes a written note of disagreement from actually making the recommended changes. One can write a suggestion, or carry it out. Having *embedded history* encourages carrying out critiques, since the original state can easily be restored. Another advantage is that in conjunction with a constraints system, impossible critiques would be prevented, because the user would try to carry out the recommendation and find that it was not appropriate. This would prevent unnecessary interruptions of other designers.

6.4 Embedded History in INDY

The primary difference between INDY and static design materials is that INDY automatically and transparently archives changes to the design artifact. INDY allows access to the artifact history via user, time-stamp, or design-unit. Currently the following access is supported in INDY:

- Find User: find where in the history transaction a user worked
- Find Different User: find where the current user started working
- Find Next Reference: find the next reference to an item in the history
- Find History Object: find the design unit referred to in a history transaction
- Find Date: find the transaction closest to a given date

Though domain independent history logs could be made to answer the question, ''What changed?'', the answer would be in terms of disk blocks. In contrast, INDY answers in the vocabulary of the domain. Design units were *created*, *deleted*, *connected*, etc.

INDY allows access not only to ''where'' but to ''when'' something happened. The motivation is to provide design information that is typically unavailable elsewhere [Hill et al. 92; Collins, Brown 88]. Figures 6-12 and 6-13 show an example of restoring a design artifact to a previous time.

These *before* and *after* images also illustrate a challenge in guiding user attention to the relevant portion of the design that is undergoing change. Even though one might notice the absence of the three workstations that have moved, one cannot tell where they went. INDY addresses this problem with two features.

First, a bird's eye view was added in to lower right-hand side of the screen to show the whole picture. (Comparing the before and after images bird's eye views, one can just make out that the three workstations were moved to the far left in the design artifact.) Second, whenever a history transaction is played, the work area is scrolled so that the design units involved are visible. The bird's eye view was not considered informative enough to indicate small changes.

One aspect of physical history that is not easily mapped to computational media is the *absence* of something. Certain deletions in the physical world are quite noticeable; they leave marks of *absence*. To illustrate with an example given in Hill etal [1992], removing pages from a parts manual can result in a conspicuous absence.

INDY communicates information about deletions in the same way it communicates other changes. Each change results in an entry in the history log. The entry is named after the verb which describes the action, e.g. *Delete Columbine*. This explicit listing of a *delete* transaction leaves much to be desired. In order to make the artifact itself communicate deletions, INDY indicates these with leaving ghost images on the screen. Figure 6-14 shows the artifact after deleting several workstations.

Having a domain specific history allows one to generate views such as showing all *deleted* workstations, or all *replaced* design units. One can thus ask questions not only about states of an artifact, but about the processes surrounding the evolution of the artifact.

A key difference between INDY and existing history mechanisms is the integration of the history with the design artifact. The integration allows the display of the history to be tailored for clarity. The history is integrated with the design artifact so that it is accessible from and has access to the artifact.

The options related to history navigation (Figure 6-16) are:

- Trace history
- Find Next Reference
- Find History Object
- Goto

Trace History brings up a menu of all past interactions related to this object. An example is shown in Figure 6-17. Also, related history transactions are highlighted. The data on each line is action, user, and date. Choosing a transaction from this menu will restore the artifact to that point in time.

Find Next Reference highlights the next reference to the design unit referred to in this transaction and scrolls the history window to show it. Figure 6-18 shows that this is similar to doing a *Trace History* and selecting the very next history transaction.

Find History Object connects the history list to the work area. Choosing this option flashes the design unit in the work area. The work area is scrolled if necessary to bring the design unit into view.

Goto plays the history to this point in time. This is the same as clicking on a history transaction.

6.5 INDY Collaboration

Embedded communication serves collaborating designers in several ways. First it integrates designing and talking about the design. This clarifies tacit knowledge by placing communication in the context in which it was elicited. INDY supports this via textual and graphical annotations which can be managed by scaling, shrinking and hiding.

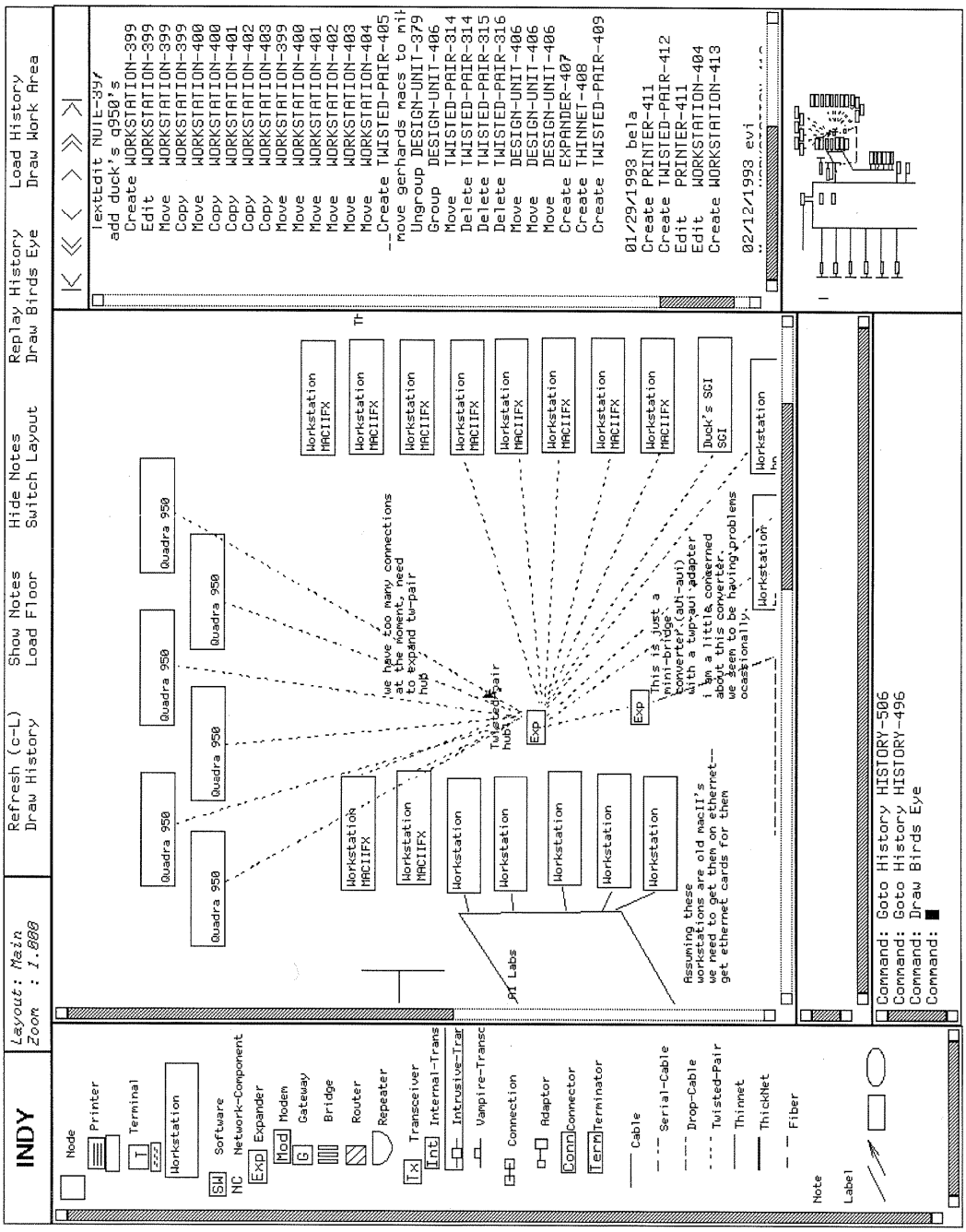


Figure 6-12: INDY View of *before* Image

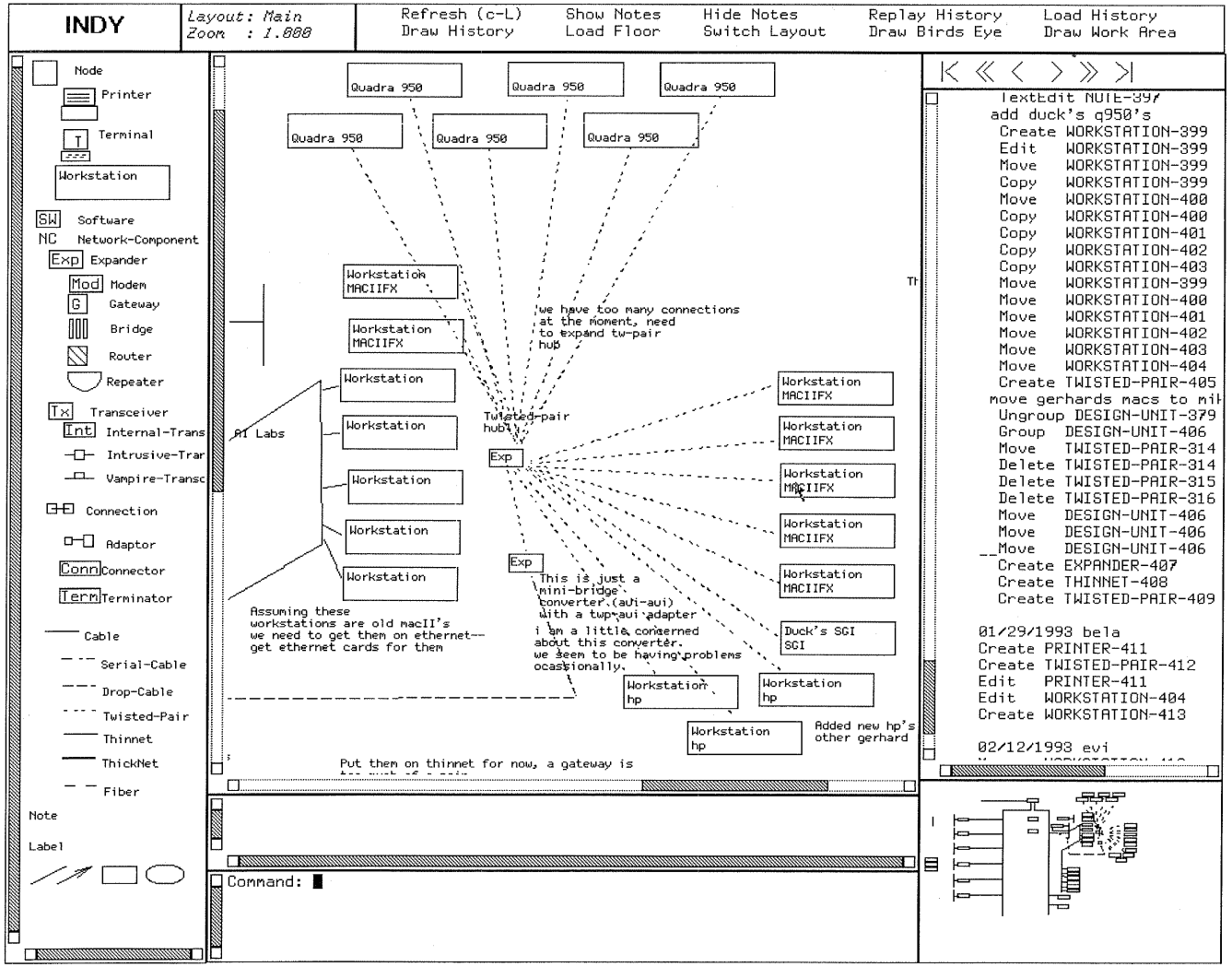


Figure 6-13: INDY View of *after* Image

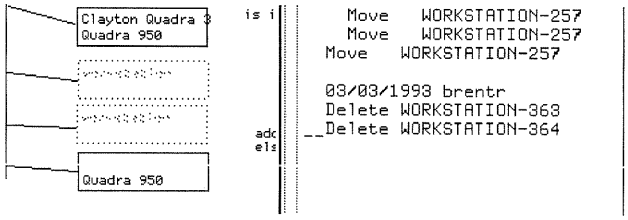


Figure 6-14: Ghost Images of deleted Workstations

INDY communicates the absence of a design unit with a ghost image.

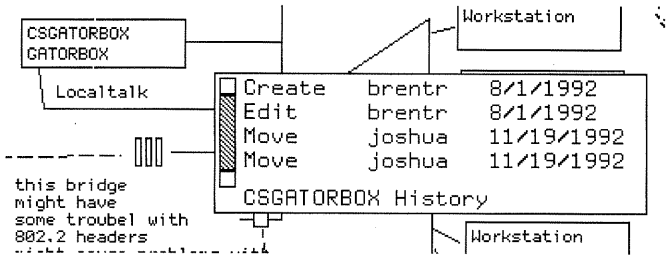


Figure 6-15: Tracing the History of a Design Unit

The history trace of CSGATORBOX

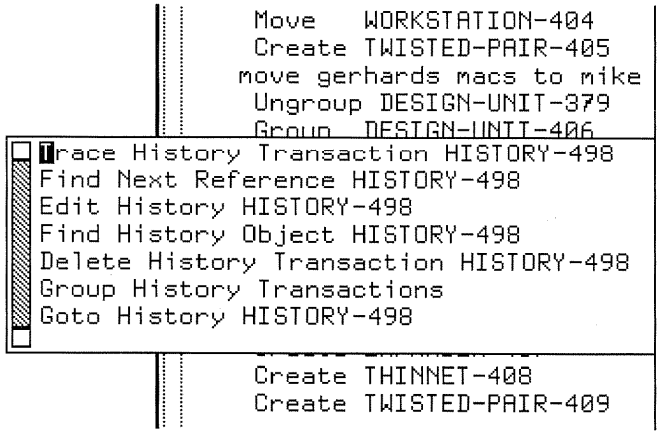


Figure 6-16: History Transaction Menu

This menu is available from any history transaction.

Though the *embedded history* is supposed to support collaboration tacitly, there are a few explicit aspects. User id's and names are kept with each design action that affects the artifact. Though this excludes such actions as changing the scaling, or scrolling.

Embedded history serves collaboration by providing a variation on *WISIWYS* (what I see is what you see), namely *WYSIWIS* (what you see is what I *saw*). Long after a designer has left the team, INDY helps one to understand previous decisions by restoring the context in which they were made. Since problem framing and problem solving are so inextricably linked [Rittel 84; Schoen 83], at any one point in time the context will play a major role in influencing the possible worlds that designers are aware of. Allowing sub-

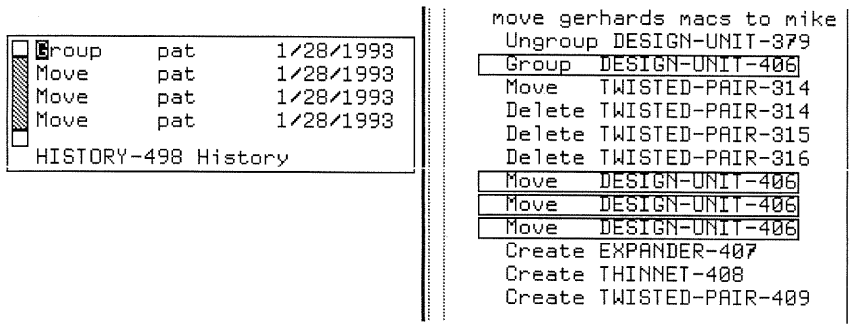


Figure 6-17: Design unit Trace from History

Tracing a design unit starting from a history transaction.

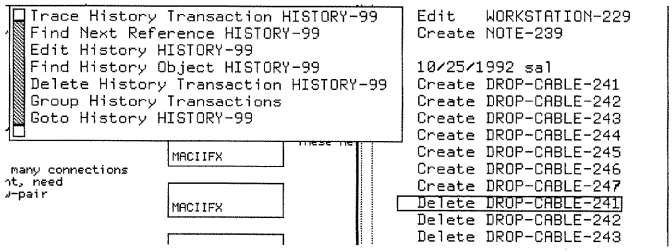


Figure 6-18: Finding the Next Reference

Choosing *Find Next Reference* is the quickest way to see where in the history this item was referenced.

sequent designers access to the context of previous decisions allows them to understand the decisions by having access to the possibilities at the time the decision was made. So this serves the task of understanding a particular decision.

Another way that *embedded history* serves collaboration is more general. Just having access to the evolution of a design artifact can provide information ''unavailable elsewhere'' [Hill et al. 92]. In the same way computational wear provides cues to understanding [Hill et al. 92], computational history helps users see patterns and draw inferences. Just seeing how something came to be can help one understand why it is in a given state.

Embedded history works in a subtle way to decrease the amount of required communication *about* an artifact. When users share a common, static artifact, it can be difficult to notice what changes have been made. Users are forced to be responsible for explicitly indicating the differences since the last version.

This has two problems. First, if a user forgets to indicate changes that affect the design in subtle ways, it can cause frustration down the line for collaborators operating under wrong assumptions. Secondly, being forced to think about the task at hand and how one will explain it to others, interferes with the very task. INDY captures changes with no penalty to the designer, who can concentrate fully on designing rather than thinking about how to describe the changes. The responsibility for communication of the changes is shifted from the original designer to the person interested in knowing about the changes. This solves the difficult problem of asking a user to do extra work from which that user does not profit. INDY requires no extra work beyond designing, but provides documentation of the changes to any inquirers.

User evaluations showed the need for higher-level abstractions than individual actions such as *move*, *create*, etc. Designers wanted to group sets of transactions and describe them so that they did not have to look at a long list of changes as remember what they accomplished. To provide a feel for the ''size'' of a group, the number of transactions is shown. For example in Figure 6-19, it took 16 transactions to ''add duck's q950's'' (Apple Quadra 950s).

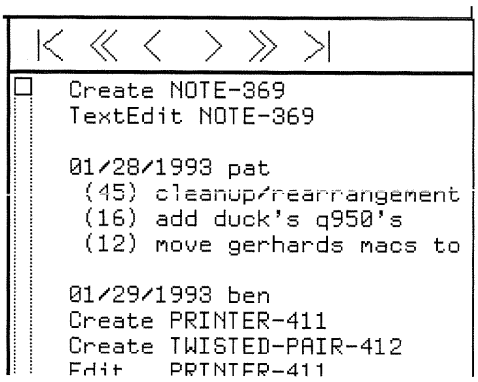


Figure 6-19: Grouped History Transactions

A major use of collaboration technology is as a reflexive tool [Thimbleby, Anderson, Witten 90]. Observations of network designers using INDY showed that history serves the task of reminding in two ways. First, the static image of the artifact can be restored to any previous date. But a static image of a previous scene does not always provide enough cues to trigger the reminding. Secondly, and more interestingly, the user can *watch* himself make changes in the past. Watching the process and not just the product of previous changes is a powerful way to be reminded of context and reasoning. This stresses the active sense of the verb *reminding* over the passive sense of the noun *reminder*. Reminding happens as a previous process unfolds.

In some sense, computer-based environments succeed to the degree that they increase the user's understanding. Though research has been done on many aspects of this and discovered the need for things like interreferential input/output, direct manipulation, active systems, etc, little work has been done in building components that integrate the history of an object with its representation.

The point of INDY's *embedded history* is not that it is superior to other methods for building understanding, but that it is one way with which we can leverage computational media over traditional media. *Embedded history* increases understanding by not only clearly portraying the important aspects of a given artifact, but by also showing *how* the artifact came to be.

When a user reviews his own work, seeing this process can help him remember the context and thus increase his understanding of previous motivations and design rationale. The history serves to remind, which serves to help understanding.

When a user reviews another's work, seeing the process is better than just seeing before- and after-images. The sequence of design moves indicates commitments in certain directions and also shows intermediate steps which do not show up in the after-image. As mentioned above, INDY allows users to adjust the rate of change.

When a user reviews a long sequence of changes by several users, it helps to illustrate the differences of opinion that took place. Seeing the relationship of one user's work to another from a global perspective can highlight interesting attributes of the design artifact, such as which area has undergone the most change or areas of primary responsibility for certain individuals.

6.6 Summary: Mute Buildings and Talking Designers

Using dynamic media is one step towards addressing the problem of *mute design artifacts* [Rittel 84]. The replaying of the artifact history tells the story of how this artifact came to be. Supporting designers' conversations with materials and collaborating designers also serves to bring artifacts more to life. The history of arguments and critiques, of blind alleys and mistakes all contribute to help current designers make more sense of the artifact designs they inherit and become responsible for. Additionally, embedding communication and history in artifacts supports the current designers in their current work.

7. INDY User Evaluation

7.1 Introduction

Users were involved with INDY in two ways. First, as described in Section 5, they helped in the design process by evaluating early prototypes. Second, they participated in a long term study of collaborative design. This study took place over a 5 month period and involved two designers taking turns designing and evaluating each other's work, followed by others analyzing the resulting design. It was conducted in a participatory design fashion and user suggestions guided the implementation. The purpose of the long term study was to simulate the process of several project members collaborating asynchronously and to evaluate the role played by embedded communication and embedded history.

7.2 INDY

INDY development began with physical views, 2d floor plans of the engineering center. Initial testing of the physical view motivated an interface to AUTOCAD files and design unit scaling. The facilities maintenance department at the University of Colorado has AUTOCAD files of the entire campus. An interface was written to import these into INDY. Evaluating the physical view caused adjustment to the scale of design units so that they were appropriately sized to the default AUTOCAD scale.

But the physical view played a smaller role compared to the logical view, and even though one network designer spent time evaluating the physical view aspect of INDY, little further work was done on that aspect of the system. The rest of evaluations involved the logical view. Suggestions included palette hierarchy changes, reusable sub-components, and a modifiable palette.

Palette hierarchy changes. The palette underwent several iterations of change. The primary lesson learned was that the object language level, in this case CLOS, should be separated from the knowledge representation level. Though not difficult, each iteration of change was still tedious. This confirmed how important it is to provide users with access for adding to the functionality of the system [Girgensohn 92].

Reusable sub-components. Rather than having each palette icon represent one design unit, users wanted to group several items together to make sub-components.

Modifiable palette. Beyond allowing additions to the palette, which was easy to support, users wanted to be able to modify attributes of objects. Though a layer of *flexible objects* was added which was intended to eventually support this, it was not evaluated.

7.3 Method

Since the CSCW focus in this research motivates support for asynchronous communication, a study was designed in which two network design experts collaborated over a period of six months, taking turns designing and evaluating each other's work. During this time the system was being rewritten to support requests as they arose in the context of the work. Rather than strict user testing, the process was one of participatory design.

All participants in this study were experienced in network design, and all five had jobs in various roles of network design at the time of the study. Table 7-1 shows that experience ranged from 4 to 12 years. This included experience in systems administration and local and wide-area network design.

Initial Setup. This author entered the office tower (OT) and classroom wing (CR) backbones and the labs which are attached into INDY. To give a flavor the "size" of the task, the history list shows that 56 transactions were used to setup the initial network configuration. That is, 56 design changes such as create, move, connect were used to seed the environment in which the users tasks were then carried out.

Tasks. After a representation of the current computer network was made in INDY, two participants were given the tasks of adding workstations to this network and evaluating each other's work. This collaboration took place over a period of 5 months. User names have been changed to preserve anonymity. The

Table 7-1: Users experience and background

User	Network Design Experience
Joshua	7 years
Luke	10 years
Ben	12 years
Micah	4 years
Pat	6 years

following sequence was chosen to closely mirror current events and provide opportunities for studying collaborative design:

This author put in the office tower (OT) and classroom wing (CR) backbones and the labs which are attached.

Joshua used an AUTOCAD file to represent the Symbolics network in CR 0-20.

Joshua added 7 Macintosh computers to the network.

Luke evaluated the placement of the newly added macs, added 6 more macs and 3 SGI machines to the Human Computer Communication lab.

Joshua evaluated the placement of the 6 macs and 3 SGI machines and added 6 Quadras for Clayton's group.

Luke evaluated the current layout, added 3 more Quadra's to Clayton's group and 3 HP's to Gerhard's group.

Joshua evaluated the layout, added 6 Quadras for Mike Eisenberg and moved 3 of Gerhard's Quadras to Mike Mozer's lab.

Micah evaluated the layout

Ben evaluated the layout

Pat evaluated the layout

These tasks were chosen to coincide with recent and proposed machine purchases and changes in the Computer Science department. The tasks were close enough to the real situation to cause designers to take into account the real situation in carrying out the tasks. Five of the sessions were videotaped and protocol analyses showed an intermixing of references to the INDY representation and the CU network in resolving design decisions and evaluating the network layouts.

At the end of the study, participants answered a survey about the merits of various features of INDY. Table 7-3 shows a summary of the responses. The most well received aspect was history. The worst critique related to the interface. INDY was implemented in version 1.0 and 1.1 of CLIM and so was most closely related to the Genera dynamic windows interfaces. This caused problems since most users were familiar with X-Windows and Mac interfaces.

7.4 Results and Discussion

After the initial creation of the artifact with 56 transactions, participants used another 249 transactions to accomplish the tasks during the study. Table 7-2 shows how many notes were created relative to all other design units created. Joshua created 52 of 73 design units and 12 of the 16 notes; approximately 1 note for every 4 design units. Luke created 19 of 73 design units and 4 of 16 notes; approximately 1 note for every 5 design units.

Users had no difficulty with the asynchronous aspect of collaborating in INDY. No statements were made

Table 7-2: Creation of notes and design units

User	Date	Changes	Design Units	Notes	Deixis
Joshua	10/19/92	21	9	1	0
Luke	10/25/92	49	14	1	1
Joshua	11/19/92	72	29	7	7
Luke	12/09/92	31	5	3	1
Joshua	01/28/93	72	14	4	2
Micah	01/29/93	4	2	0	0
Totals		249	73	16	11

indicating the need to communicate with someone face-to-face. Rather, clarification requests were made in terms of the facilities already provided in INDY. Even though users would spend time reading the annotations, and even show difficulty in interpreting them, there was never a request, even in humor, to speak directly to the person who had written the notes; no comments relating to advantages of synchronous over asynchronous communication. The asynchronous interaction style currently plays a large role in how their work is done.

Critiques were expressed as annotations. Figure 7-1 shows an example of critiquing a configuration by leaving a reminder of unfinished work.

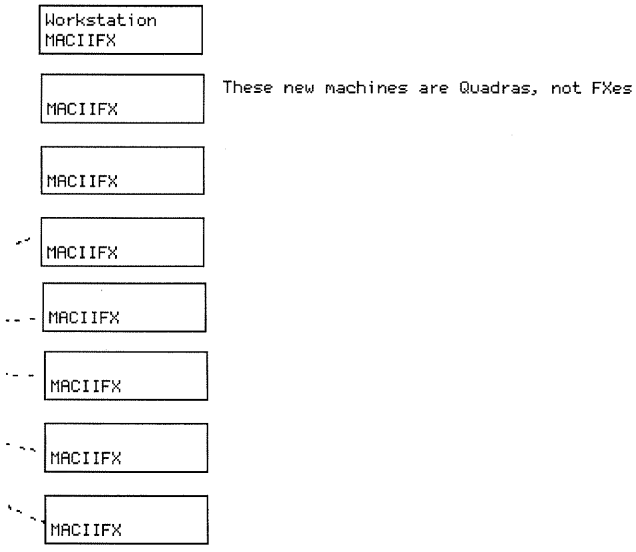


Figure 7-1: Reminder

The use of a note as reminder of something to be done.

Designers questioned the functionality of certain configurations and responded to these critiques. Figure 7-2 shows both a question and a reply to the challenge of incompatibility.

Designers did not hesitate to make changes to the artifact. This was due in part to the explicit directions to "make any changes you want to." However there was no hesitation, or indication of discomfort in carrying out that assignment. It was not possible to tell how the absence of a history mechanism would

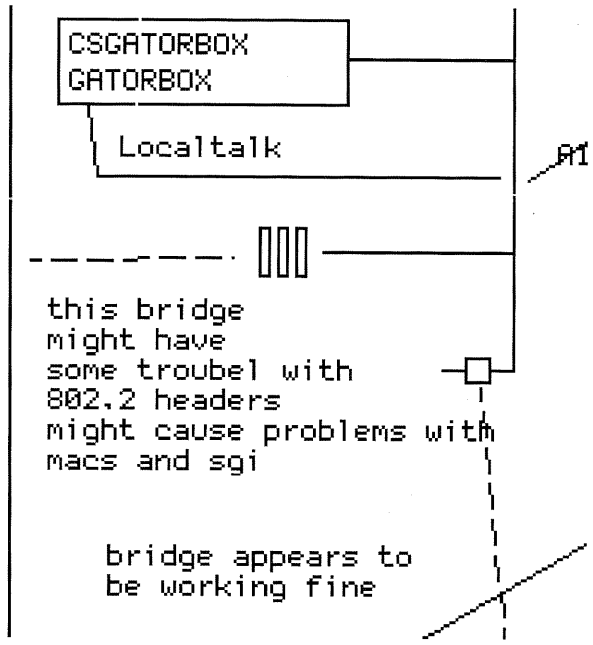


Figure 7-2: Design Issues

Notes used to raise and answer issues.

have affected this, i.e. whether the willingness to change other's work was due to the fact that the changes could easily be undone, or whether the willingness to change something was related to the individual's confidence that the changes were indeed *correct.*

In addition to positioning notes *close* to the items references, explicit connections were made in several cases, shown in Figures 7-3, 7-4, 7-5, and 7-6.

The feature that designers praised the most was history. It seems they were not thinking of it in the sense of Hollan and Stornetta [1992], i.e. archived *communication*, but rather as an archived *trace of changes.* Nevertheless one designer mentioned the time dimension of communication:

> I can see you wanting to have the notes basically, well the notes are time based, you have a time dimension. And so you might have conflicting notes attached to the same place or same object... And so you might want to see the entire sequence of notes associated with something. You might want to see the last note associated with it.

Several suggestions were made relative to history access and presentation. Users requested access to the history *by user.* This arose in a reflexive CSCW use, namely the need to find where one had done work oneself.

Nested groups were requested in response to the volume of history transactions. Scaling issues surfaced in two ways. As time progressed, even relatively minor sets of changes were forgotten; i.e. looking at a series of history transactions did not help to understand what had happened. On the other hand, as the size of the set of transactions required to do a semantical unit of processing increased, it also became impossible to understand from looking at the list of transactions, what had happened. The history list was modified to be a hierarchy, so that arbitrary levels of nesting are supported.

Users suggested that "wiring" email be integrated with the design units. The computer operations group (CSOps) maintains several email logs which serve certain functions of network maintenance. One of these logs is called *wiring* and is the repository for email related to connectivity issues and running cables. Another log is called *diary* and contains logs of changes to machines such as software and hardware upgrades. The suggestion was made that INDY notes should be integrated with these email logs in such a way as to access all notes related to a given workstation or subnet.

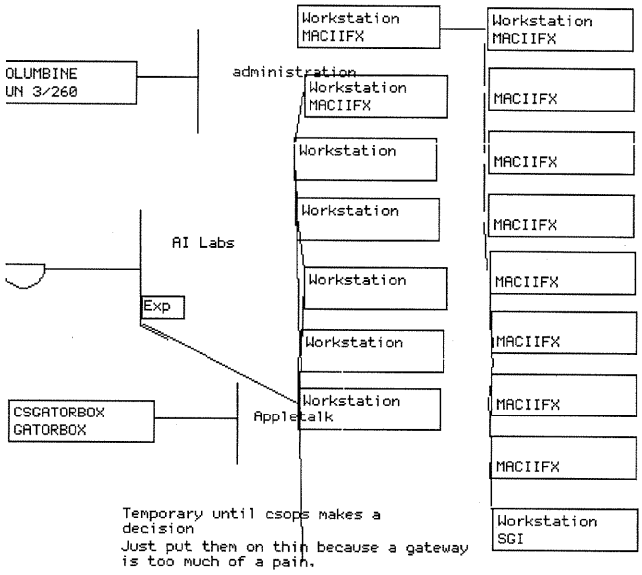


Figure 7-3: Connecting Notes to Design Units - 1

Though not shown, a logical connection was added connecting the bottom note and the SGI workstation.

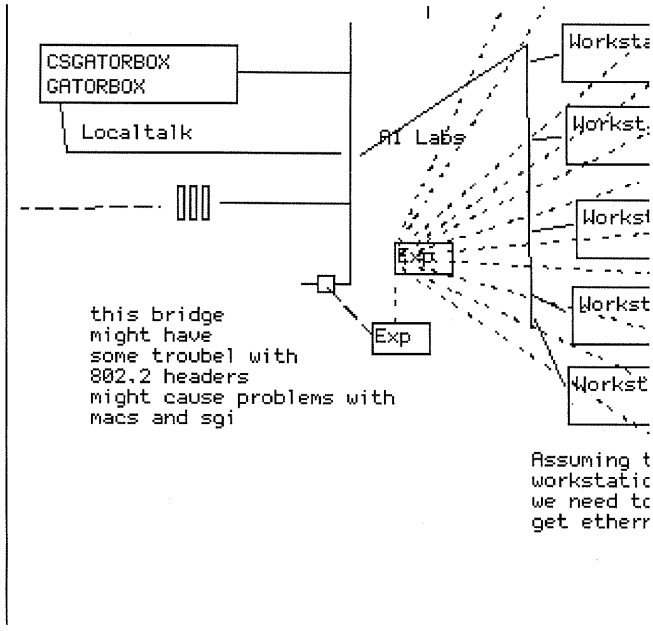


Figure 7-4: Connecting Notes to Design Units - 2

The note was created on 11/19/92 (context shown here), but not logically connected to the bridge until 01/28/93 shown in Figure 7-5.

The integration of history and artifact was well exploited. The following quotes from the transcript motivate INDY's history access mechanisms:

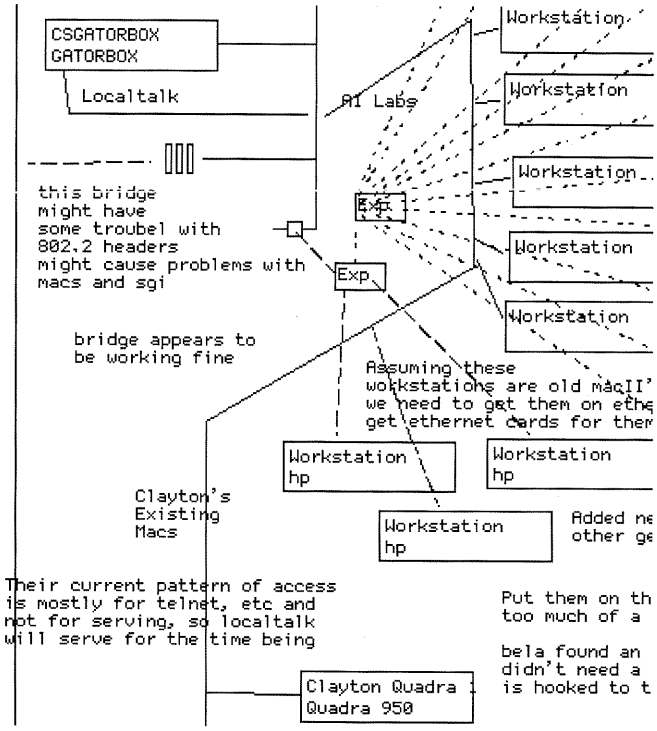


Figure 7-5: Connecting Notes to Design Units - 3

Long after the note was created, it is logically connected to a design unit.

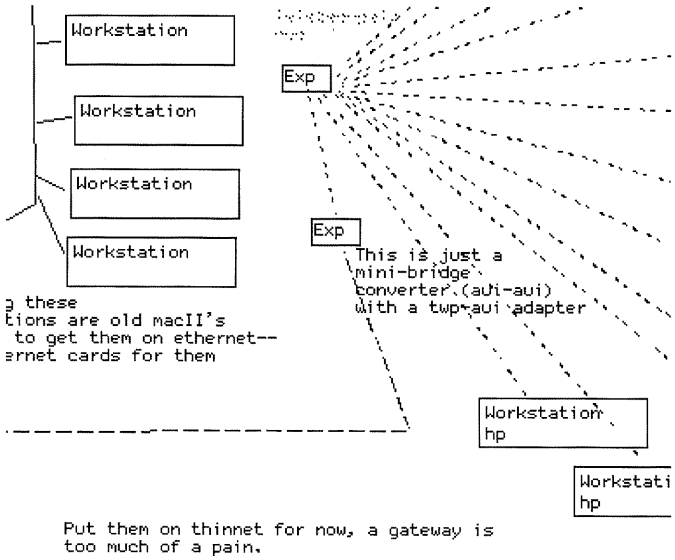


Figure 7-6: Connecting Notes to Design Units - 4

I don't remember exactly where I was last time. Is there a way to determine where I left off?

Sometimes it'd be useful to know who made this change in the design.

Another thing that might be useful would be for me to have some way to say ''Where was that created and who created it?''

So is there any way to find out who made these notes?

Is there a way to find the history of an item?

But that's incorrect... you see here, history's useful, because I want to go back to the context of when... I started more or less...

These comments and questions motivated the specific commands described in Chapter 6 and confirmed the role embedded history plays in understanding design artifacts that have evolved over time.

7.4.1 Missed Opportunities

Certain things happened during the study which are here labelled as ''missed opportunities''. This was when this author as the system designer knew there was a functionality in the system which would serve the user in a given task, but which the user did not exploit. One could not always be sure of the reason; some were caused due to the complexity of the system and the difficulty participants faced in remembering features which were not available in any of their usual design and maintenance tools. Other times it might have been that the users did not perceive a given feature as useful for their purposes.

No designers scaled individual design units or notes in order to clarify relationships. They did however arrange notes ''close to'' the items they referred to. But the reference was not always understood. Notes which were originally placed *near* certain objects turn out to have been *near* only in a relative meaning. As other areas of the artifact grow, what used to be near to one thing can suddenly be *nearer* something else. The tacit meaning of spatial positioning is lost. Figure 7-7 illustrates how important context is to understanding. In the note ''Put them on thinnet for now'', what does the ''them'' refer to? The three rectangles labeled ''Workstation'', the column of machines to the right starting with ''Duck's SGI'', or the column of machines on the left bottom starting with ''Clayton Quadra''? At the time the note was written, it was unambiguous. By restoring the context when the note was made, however, references were clearer.

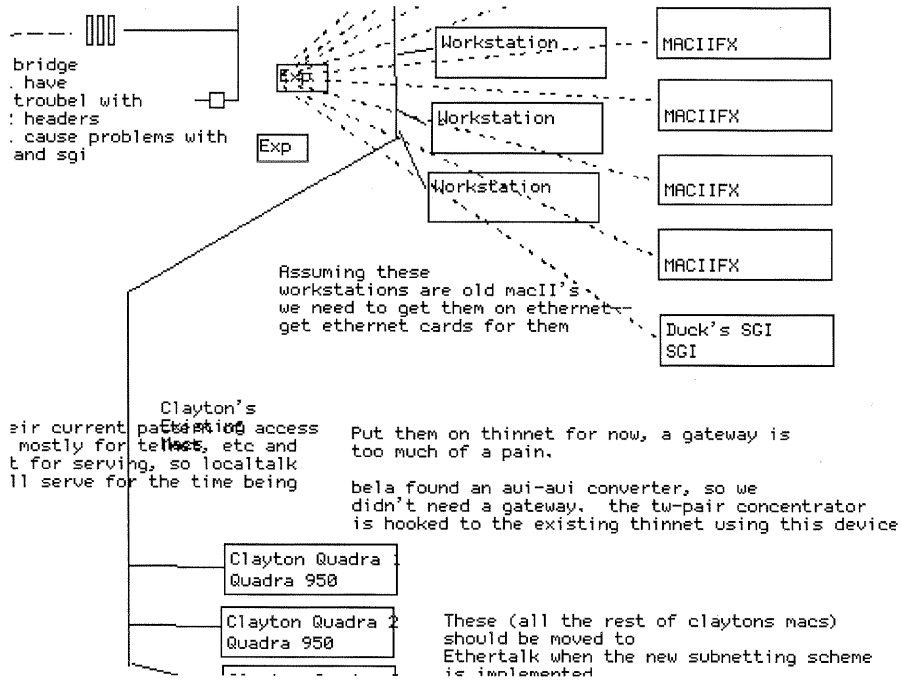


Figure 7-7: Meaning and Context

Artifact growth obscures tacit knowledge. In the note ''Put them on thinnet for now'', to which design units does ''them'' refer?

No designers used sketches such as lines and ellipses to group or draw connections between items. This might have been due to the fact that they were introduced after the study had begun. They did however use the ''logical'' *connect* to associate things. By doing a *show connections*, one could get a list of connected design units. Having the connection shown graphically instead of just represented logically would have clarified references at less cost to the reader.

The tasks were purposely written down on a piece of paper, rather than using textual notes to embed the tasks in the artifact. The reason was to see whether anyone would decide to enter the tasks, or whether the absence of the task in the artifact would ever get noticed. The following dialog shows how it did surface[4]:

D: trying to figure out what these machines are. The information I don't seem to have is, I don't know what these machines are. These are probably 1,2,3...9 quadras, so I know, I have in mind instances that match those.

M: Ok

D: These are still too generic for me. Is it one that's been moved in already? Is it sitting over there waiting to be placed?

M: I had X place 3 SGI's, 1 for mike

D: *That information in the made up task would have been useful for me to have now*, because to me, the predominant use for this part of the net over here, this thinnet, is group Y, we do happen to have something else hanging on right now.

As expected, no designers entered any of the tasks into the design artifact. Though annotation and artifact are integrated in network design meetings on paper and whiteboard, they are rarely integrated in computational media such as MacDraw^{tm} which the network designers had used some in the past. Expectations of computational media seemed to interfere with exploring new uses.

7.4.2 New Problems

Embedding communication and history in a design artifact was certainly shown to be useful for collaboration, but certain new problems were raised.

Though communicating, in this case limited to textual annotations embedded directly in the design artifact, helps clarify design intentions and configurations, it also seems to interfere with understanding the design artifact. Simply allowing textual annotations to be integrated into the work area was not enough. Designers asked for the ability to hide/unhide notes. Keeping notes visible seemed user- as well as context-specific. Certain users preferred that they stay off, and only turned them on when they felt the need to clarify aspects of the artifact. INDY was implemented to force the 2 dimensional integration of text and graphics in ways network designers are not accustomed to seeing. By forcing this arrangement, certain difficulties were raised which are now open to analysis. Suggestions for managing notes included stacking them ''on top'' of one-another or only showing a few of the ''most recent'' ones.

Though subjects used the communication aspect, textual notes, to collaborate, this was not without cost. Several unclear references caused collaborators to do extra work to disambiguate. Though notes were used, there are difficult issues of scalability to resolve. One answer is to *Hide Notes*, and proceed. This however results in design units being placed on top of invisible notes, which when set to be visible again, *really* mess up the picture. So simply hiding notes is not the answer.

One approach is illustrated in PAD (discussed in Chapter 8), a system with infinite scalability. Perhaps the difference between *completely unseen*, and *very, very small, but still noticeable* could turn out to be the telling one. Instead of making notes completely disappear, one could just scale them so small as not to interfere, and yet at the same time, remain *slightly noticeable.*

Allowing users to delete and replace design units freely can result in lost information. Figure 7-3 above showed the context of when the note ''Just put them on thinnet...'' was logically connected to the SGI workstation. Later on, that note was deleted and rephrased in a new note, but the connection to the SGI workstation was inadvertantly lost. Figure 7-7 shows the state at the end of the user session in which the note was added. It already is ambiguous even before another user has changed the artifact. A collabora-

[4]D is for Designer, M is for this author

tive system should certainly help designers to preserve any *added value* of information that was entered. In the case of replacing a design unit that is connected to other units, the user should be made aware of how it is related to those other units, so that no information is lost.

As the study progressed, certain vocabulary surfaced that seemed to have a meaning beyond the ordinary associations. For example:

> D: Is there a way I can change the type of something? This is not really a workstation.
> M: You can delete it and create a new one.
> D: But is there a way I can say this is changed to something else?

As network equipment is updated, it is sometimes replaced, but other times it is *changed,* e.g. a rom upgrade adds new features, or a new card changes the way the device interacts with the network. The device is still the same device, but it's performance has changed. This seems mostly to be related to bridging devices such as modems, routers, bridges, repeaters, and gateways. Certain workstations can perform certain aspects of these devices, yet remain fundamentally a workstation. This raises the issues of vocabulary for describing or representing the history. If *change* has a special meaning, and consists of subtasks such as deleting or removing a part and adding another, then the history should be represented in terms of a *change* rather than a sequence of steps which accomplish this. The history should represent activity in terms of the domain workers' *language of designing* [Schoen 83].

Though this was the only discrepancy between domain vocabulary and history vocabulary that surfaced in the study, it was nevertheless a reminder of participatory design; understanding the language of the domain workers.

The issue of privacy was already mentioned in Chapter 6, where a user first considers something to be "stupid," then acknowledges that there are circumstances under which it would be perfectly acceptable. No other interactions raised this issue as clearly. However, no users ever expressed hesitation about the fact that their interactions were being archived.

7.5 Relation to Empirical Work

This section describes the relationship between the empirical work described in Chapter 5 and the long term study described above.

The two main insights from the McGuckin study were that problems and solutions co-evolve and that critiquing plays a central role in collaborative design. The first insight was not seen in the long-term study of INDY. This was due to the fact that although the tasks which were given were taken from the current network situation they nevertheless were assigned, making them less subject to reframing. The second insight however was confirmed as notes were written which critiqued aspects of the design and subsequent designers responded to these critiques. Unstructured textual annotations served this task well.

The Bridge study showed that when dealing with physical artifacts, notes can cause clutter and also become lost. Both findings were supported in the INDY study. The clutter problem was temporarily solved by hiding all notes. This in turn led to a worse problem: when the notes were turned back on, new design units had obscured them. The opportunity of scaling the notes down to be unobtrusive, but still noticeable, was not taken advantage of by the network designers.

The prototype JANUS-NOTES resulted in requests for textual notes which were free of any imposed structure and INDY thus supported free-form textual annotations. Users liked being able to write in this unrestricted fashion, and none asked for structured links.

Analysis of video-tapes of collaborating network designers surfaced the large role that deixis played. This was also born out in the user-study, as 11 of 16 notes contained deictic references. The long term study showed clearly how a history component can be essential in disambiguating deictic references.

7.6 Summary

Table 7-3: Evaluations

Question	Joshua	Luke	Ben	Micah	Pat
1 System useful? 1 not..very 7	5	na	5	4	7
2 Cf current software? 1 less..more 7	6	na	6	6	7
3 Notes useful?	6	na	2	5	6
4 Understand Notes? 1 difficult..easy 7	4	na	7	2	4
5 History useful? 1 not..very 7	6	5	4	7	"7, 7"
6 Concept? 1 useless..useful 7	6	na	5	6	7
7 Implementation? 1 useless..useful 7	3	na	4	4	7
8 What could you not do that you normally can do?	nothing	design vs documentation	color	new palette objects	na
9 What could you do that you normally cannot do?	history	history	pictures	history replay tracing annotation	history
10 More effective features?	history notes	na	pictures	added features	history, huge worksheet, finding objects
11 What did you like?	history	history notes	pictures	history and replay	sophistication
12 What did you dislike?	interface	logical connections	interface	na	na
13 What was hard to understand?	interface	can't remember	logical connections	symbols	na
14 Collaborative aspects?	na	user history	rcs style pictures	interface	decomposability

INDY is a complex system with many features. Subjects were not able to learn all the features. Each session was limited to 90 minutes and designers spent some of their time analyzing the system and making suggestions for improvements, rather than using all of their time "on task."

In the current network design group, there are currently no computer based tools in use beyond email and occasional use of MacDraw^{tm}, so just about any tool would be new. The CLIM interface is quite different from X-Windows based tools such as Motif and thus even simple things like scrolling caused trouble. This especially hindered the last three subjects brought in as "outsiders" to evaluate the work that had been going on. The features of INDY intended to support interpretation and understanding were explored and used in analyzing the existing artifact, however before completing the task of evaluating the design, subjects became sidetracked in exploring the history and notes, and ran out of time before producing critiques like the two collaborating subjects had done.

Knowledgeable domain workers guided the design and implementation of a system for local area network design. They also participated in a long-term collaborative project in the context of the University of Colorado engineering center. Analysis of the interaction shows that embedding textual notes in an artifact normally reserved only for "the artifact," leads to advantages as well as disadvantages. Hiding notes

seemed to be a personal preference, though all subjects did unhide and read them to understand the network layout.

Users really liked history. They used various access methods provided to restore context or watch sequences of design changes being reenacted. History clarified written notes by restoring the context in which a note was written. Notes which were clear when first written became ambiguous as the artifact evolved in unanticipated ways.

Embedded communication and *embedded history* served to reinforce each other. Having a history of communication places that communication in a larger context of time, rather than just a 2 dimensional context of the design artifact. Having communication in a 2 dimensional space rather than just an email list clarifies references and tacit knowledge.

Embedding communication and history in a design artifact makes more information public. Users took advantage of this by taking into account who had made a given change. But an evolving design artifact leaves certain actions open to unjustified criticism. The hope is that enough context can be presented to inquirers to prevent misinterpretations of past actions.

8. Related Work

8.1 Asynchronous Collaboration

The areas most closely related to the spirit of *embedded communication* are asynchronous communication and annotation support. Though more and more collaborative annotation tools are becoming available, most focus on synchronous communication and are therefore not included in this review.

8.1.1 NOTECARDS

NOTECARDS is a hypertext-based idea structuring system [Halasz, Moran, Trigg 87]. Though not initially designed with collaboration in mind, users did work together [Trigg, Suchman, Halasz 86]. The result of collaboration was a list of general requirements for collaborative idea structuring systems.

The aspect of NOTECARDS most closely related to this work is *mutual intelligibility*. In addition to the work itself collaborators must find ways to maintain the coherence of their interaction. Trigg, Suchman, and Halasz [1986] describe the different kinds of work that should be supported in a medium for collaboration:

> Substantive activities are those that constitute the work at hand. When writing a paper, the substance of the work is the ideas expressed in the paper's actual text. In design the substance is the plan and implementation of the artifact, in research discussions the problem or topic of interest...
>
> Annotative activities are *about* the work and include commenting, critiquing, questioning and otherwise annotating the work itself...
>
> Procedural activities include discussions about conventions for use of the medium or technology, logistics of turn-taking, record keeping, etc.

The idea of different kinds of communication taking place with one medium is also related to Schoen's observation of language *of* design and language *about* design both being intertwined in design discussions.

My original interest in this topic was guided by the suggestion that:

> In order for people to collaborate, they must be able to make their work mutually intelligible. This requires annotative and procedural as well as substantive work. Ideally, computer systems designed to support collaboration should capture both sorts of discussions and store them in the same medium as the work itself [Trigg, Suchman, Halasz 86].

8.1.2 PREP

PREP is a system that supports loosely coupled collaborating authors [Cavalier et al. 91; Neuwirth et al. 90; Neuwirth, Kaufer, Chimera, Gillespie 87]. The goal of project is a multi-user environment to support a variety of co-authoring and commenting relationships for scholarly communication. The PREP text editor supports social interaction among co-authors and commentators as well as cognitive aspects of co-authoring and external commenting. It emphasizes communication, planning, and organized annotation.

In line with *embedded communication*, a central focus in PREP was on usable visual representations that support new kinds of communication in addition to traditional styles of collaborative text editing. The document is organized by columns, which can be used for different grouping purposes, such as the text itself, plans, outlines, or various author's comments.

Though columnarizing a document to allow for collaboration has intuitive appeal (the idea was inspired by glossed bibles in which space was left just for comments and elaborations), there are limits to the approach. As soon as more than a few authors participate, it becomes difficult to manage the separate columns, most of which are empty most of the time. The column approach suffers from the same limitation that spreadsheets suffer: when there are many comments (spreadsheet macros) most of the "real estate" is wasted. The reason in spreadsheets is that any row or column is subject to deletion, so macros

must be placed in rectangles "below, and to the right" of the numbers. So even though many rows and columns are taken up, most of the cells are empty.

Second generation spreadsheets addressed this problem by supporting named macros which could reside anywhere in the work space and not be subject to the user deleting a row or column. But to do so, they had to break with the basic metaphor of infinite rows and columns.

Overcoming the column problem is certainly possible, but you must break with the basic metaphor, e.g. allowing multiple authors to comment in the same column, or collapsing intermediate empty columns.

8.1.3 FREESTYLE

Collaborative work involves sharing artifacts which are annotated with critiques. Wang's FREESTYLE is a product that supports annotation, electronic, and voice mail [Francik, Rudman, Cooper, Levine 91]. A person using an MSDOS machine can "grab" any screen, say a Lotus spreadsheet, and mail it electronically. In addition to mailing any screen, one can also annotate it in two ways: 1) by use of a stylus, e.g. drawing circles around ''important'' items, and 2) by voice mail. The receiver of the message gets not just the static application screen, but a dynamic replay of the voice mail and the stylus gestures that the sender made.

Though voice data might be easy to produce, it is difficult to navigate. There is no way to scan it quickly to get a feel for the content of a message.

The FREESTYLE system is domain independent in that it can grab any screen that can be produced on an MSDOS machine. This limits the possibilities for annotation. There is only a snapshot of the application, a static representation of the state of the program. In contrast, the approach suggested here is to integrate the critiquing component into the design environment so that anything that can be done in the design environment can also be done to illustrate a critique. The purpose is to keep the designer from having to think about the limits of an annotation system, and instead, just continuing to work within the familiar design context.

8.1.4 Word Processors

Though the generic category of *work processors* covers a wide range of functionality, there are nevertheless features at even this general a level which support communicating within artifacts.

Change bars draw readers' attention to places certain places in the text where allegedly something has changed. This paragraph is marked with change bars.

Conditional Text. Systems such as FRAME support an idea from hypertext which allows text to be conditionally displayed. The interface to this functionality varies, but can include user-named conditions, color, fonts, etc to characterize authors or sections of a document. One can hide/unhide these to produce various versions of the printed document. Hidden text provides a way for authors to annotate the artifact without affecting the printed version, yet one can also print the version with all comments *turned on.*

8.1.5 PAD

PAD is a system for sketching and writing on a virtual 2d surface that provides infinite zooming [Perlin 90]. The basic idea is that all data and programs on a shared computer network reside at specific geographic locations on an infinitely detailed two dimensional data surface.

There are no windows as in traditional graphical interfaces, so there is only one view of the data. However one's view can be rescaled arbitrarily to go rapidly from global- to detailed information.

The interesting view that PAD brings to this thesis is the idea of an infinitely scalable artifact medium. If collaborating on a text document, the pages can be laid out say from left to right on the work space. Annotations could be scaled small enough to fit between pages or even columns in a two-columned paper. When printed, the resolution of the printer would make the annotations invisible.

The potential of scaling annotations so small that they don't obscure the artifact also presents the problem of finding them. Infinite scalability introduces navigation problems similar to hypertext [Marshall, Irish 89].

8.1.6 OBJECTLENS

Structured Communication leverages informal electronic communication by adding structure to messages [Lai, Malone 88]. The idea is to automate much of the standardized message processing that is part of collaborating group's interaction. An example is OBJECTLENS, an electronic communication structuring tool [Lai, Malone 88]. It is the "second generation" of the Information Lens [Malone, Grant, Turbak 86] system and adds a large number of enhancements, including ideas from artificial intelligence to improve the kinds of knowledge that can be represented. Users can define and modify templates for semistructured objects, and create semiautonomous agents to automatically process information contained in the group communication.

8.1.7 Summary

Asynchronous communication systems have focussed on the communication aspects and not considered the context in which the tools serve. Since collaborating designers do not artificially separate communication and design, the tools built for them should integrate these aspects.

8.2 History Tools

8.2.1 Taxonomy of History Tools

Lee [1991] provides a summary of recent work in history tools and identifies seven uses of history:

1. reuse: reusing scripts directly or with modification
2. inter-referential I/O: referring to previously displayed information in current interaction
3. error recovery: undo mistakes
4. navigation: finding out one's current place, and path taken
5. reminding: aiding recall and providing visual cues to past events
6. user modelling: responding to differences in individual users
7. user interface adaptation

Studies of the use of history have focussed on history as a domain- and application-system independent tool for user support [Lee 92b; Barnes, Bovey 86]. User activities contain repetition, which motivates a way to provide support for easily reusing previous commands [GreenbergWitten 88]. The area most researched and most supported by computer-systems is *reuse*. According to Linxi and Habermann [1986], the most common use of a history tool is to reuse and possibly modify a history item to save keystrokes and/or mouse strokes. Though undoubtedly useful in simple tasks such as UNIX command line submission, focusing on reuse of an operating system command overlooks how people use computer-based environments to carry out complex design projects.

Though these interactions with an operating system are important and tools such as TCSH (discussed later) are a useful addition to operating system interface, the outcome of a multi-year design project hardly depends on whether the designer types *lpr project.ps* or reuses a previous *lpr* command. Whether it is command-line recall, or macro recording, the support is still barely above the keystroke level [Ellise, Greer, Placeway, Zachariassen 87; Stallman 81]. Few would argue the usefulness of macros, but higher levels of support are needed for collaborative design.

This taxonomy assumes a history is user-specific. However in collaborative design, an additional

perspective is helpful, that of artifact-specific history. The focus shifts from the individual to the shared artifact.

8.2.2 Version Control Systems

RCS is an example of a version control system and manages changes to text files, typically source code [Tichy 82]. The primary use is in software version management. RCS helps users to manage changes to text files by tracking *revisions*. To make changes to a file, one checks it out, edits it, and checks it back in. RCS tracks the changes in delta files, similar to the unix *diff* command. RCS can handle multiple branches, based on the idea of software version numbers such as ''Release 1.3'' vs ''Release 2.0''. This lets two programming teams use the same text database to develop different versions.

Though necessary with projects of any significant size, the level at which one would like to manage complex software is not at the source-code-line level. However managing software at anything but the source code level has proven difficult. Domains in which graphical representations play a larger role, such as network design, allow us to explore history mechanisms at a higher level. The hope is that insights gained from this kind of domain will shed light on how better to represent the history of artifacts which are mostly represented by text.

8.2.3 Command Line History

TCSH is a UNIX C shell with file name completion and command line editing [Ellise, Greer, Placeway, Zachariassen 87]. It is representative of many command line history tools such as C Shell [Joy 87], Interlisp-D [Teitelman, Masinter 81], and Symbolics [McMahon" 87].

Tcsh provides easy access to the history of commands. Navigation keystrokes like ^P and ^N display the previous or next command in the history sequence. Pressing <enter> after recalling a command enters it to the operating system. After a command has been recalled, it shows up in the current line and the cursor is placed at the end of the line. The user can now use editing commands such as M-F (forward word) or ^D (delete character) to easily change parameters of the command.

Search access to the history is provided by matching characters starting at the beginning of the line. For example "echo<ESC>p" would find the most recent command starting "echo".

TCSH is a good example of providing extra functionality at no cost to the user. One can be running TCSH and not know it; unix commands work as expected and there is no user-observable penalty for running tcsh. However with very little effort, one can reuse previous commands and get filename completion. The threshold for using it is very low.

8.2.4 Interreferential I/O

MACSYMA is a symbolic algebra system that allows references to previous expressions to be incorporated into the current input. The interaction with macsyma is a read/eval/print loop. The user enters an expression which macsyma attempts to evaluate. If the evaluation is successful, a result is printed prefixed with a unique line number. The user can then incorporate this line number in subsequent input to refer to the whole output expression. Reuse is via a unique reference number which thus does not carry over from session to session.

The Genera Presentation Type system is a user-interface substrate that facilitates reusing output from previous interactions as input for current dialogs [McMahon" 87]. The substrate is intended for programmers writing direct-manipulation interfaces. As in MACSYMA, users can include previously displayed objects in current input, but they can do so simply by clicking the pointer on any screen object that will ''highlight''. The input is context sensitive and disallows objects that are of the wrong ''type''. It is a non-trivial task to design presentation types, but once defined, they work across all input and output contexts.

Though not intended strictly as a history mechanism, presentation types do support history. Output that

has scrolled off the screen can still be re-used. One scrolls to the displayed object and it will be mouse-sensitive as though it were on the current screen.

8.2.5 Group Sketch tools with History Support

We-Met (Window Environment-Meeting Enhancement Tools) is a prototype pen-based tool designed to support both the communication and information retrieval needs of small group meetings [Wolf, Rhyne 92]. We-Met supports small, informal work groups by capturing sketches done during the meeting for later time-based retrieval.

This work-group support system tries to overcome some of the limitations of video tape as meeting capture medium. It supports time-stamped history of meeting events. Users can scroll through the history and watch as sketches get created. Certain events in the history may be marked by textual or sketch annotations. This facilitates later information retrieval.

8.2.6 Read/Edit Wear

As argued in the Chapter 2, one view of design is as a ''reflective conversation with work materials'' [Schoen 87]. In an attempt to provide computer support for this ''conversation'', Hill et.al. [1992] have explored an informational physics perspective on interface design in two applications: reading and editing.

The *wear* in *Read Wear* and *Edit Wear* refers to the tendency of the use of an object to lead to noticeable wear and tear. Paperback books are a good example of how use shows. If you close a paperback book without marking the page, then come back later, it will open to about the same place. The physical material changed according to use and Hill argues that changes like these can be useful to humans; that emulating these kinds of changes in computational media provides information otherwise unavailable.

Representing usage wear computationally helps answer questions like, ''Which sections of the document are most stable/unstable?'' or ''What are the relative ages of document sections?''. *Edit wear* also tracks information to answer questions like ''Who wrote what?'' and ''Who edited what and when did they edit it?''

One question which edit and read wear cannot answer is, ''How did the artifact look when user *x* changed *this*?''. Though one can see evidence of the history of an artifact, one cannot see it as it was. One can ask ''When did it happen'', but not ''what did it look like *then*?''. You can go to *where* it happened, but not to *when* it happened. Though an improvement over more typical computer-based representations, the wear metaphor is not enough to support long-term collaborative design.

Another shortcoming of portraying computational wear is that deletions are not depicted, yet in the physical realm, if enough pages are torn out of a parts manual, it *is* noticeable. There needs to be a way of portraying the deletion or absence of a part of the artifact.

8.2.7 Summary

The main shortcoming of systems which provide history tools is that they do not integrate the history tool enough with the design task. To provide the kinds of services designers need from a history, that history component must have access to the underlying representations in the system.

9. Summary and Conclusions

9.1 Summary

This work explores tools for asynchronously collaborating designers and rests on three primary resources. First is Schoen's theory of design and his metaphor of design as a conversation with the materials of a situation. Schoen's conversation metaphor has been extended as follows. The *conversation* that takes place between designer and materials is broadened to include the conversation between collaborating designers. Schoen's *materials* are extended to become dynamic computational media. Schoen's *situation* is extended to include the historical as well as the current context.

Second is research in cooperative problem solving systems which has led to the development of domain oriented design environments. Their emphasis on providing computational support for designers, along with the views of design as drawing, constructing, and arguing issues, provide a framework for instantiation of a system to support collaborative design.

The third resource is computer supported cooperative work which reminds us that collaborative design is primarily about people communicating with people. Asynchronous systems such as electronic mail are sometimes used even when synchronous access is possible.

Researchers in collaborative systems have explored all quadrants of the time×place matrix in Figure 2-1. This dissertation has explored the different-time/different-place quadrant, arguing that asynchronous communication has much to offer collaborating designers.

These three resources motivate a view of collaborative design as "communication over time." People share artifacts and communication and the two are intertwined. This motivates computational support for integrating these two aspects of the design process.

Empirical studies of people collaborating to solve problems, design toy bridges, and design networks, raised issues about the open-ended nature of problem-solving, the role of critiquing in design, the reliance of humans on deixis in communication, and the use of history in understanding an existing design.

The McGuckin study showed the role of critiquing and the co-evolution of problems and solutions. The interaction between customers and sales agents could often be described as critiquing. Both participants made suggestions, corrected the other, and asked for help in brainstorming. Customers and sales agents worked within problem and solution spaces simultaneously. Typically the problem owner had a better grasp of the problem space and the problem solver had a better understanding of the solution space, but over time these spaces converged until there was a large enough intersection of shared knowledge within which potential solutions could be evaluated.

The Bridge study raised issues relating to embedded annotations and design history. Textual annotations can obscure the design artifact. They can also get lost entirely. Designers expressed a need for a design history, which would show how the artifact evolved. Design goals were not obvious and were only discovered through a process of critiquing and evaluating.

The Network Design video showed the role of deixis and sketching. Over one fifth of the sentences spoken during a design session with collaborating network designers contained references to visual representations. These (audio) references could not be disambiguated without access to the visual context (the video).

Finally, the JANUS-NOTES systems provided the first user feedback about textual annotations. Users suggested ways to manage annotations by scaling and hiding. They also suggested ways to structure collections of notes into discussions.

These studies guided the implementation of INDY, a system which supports the collaborative design of computer networks. The two major contributions of the INDY system are *embedded communication* and *embedded history*.

Though embedding communication about an artifact in that artifact provides collaborating designers with certain benefits, there are also drawbacks. A long term study of collaborating network designers showed that textual annotations can add clutter to the artifact and so make it more difficult to interpret. Embed-

ding notes in the two dimensional space of the design artifact resolves ambiguous references in the communication and places it where it is most useful. But this embedding also leads to clutter and requires a way to manage the notes.

Embedded history provides great benefit at very little cost to the user. Even when scaled up to complex multi-year projects with many designers, the approach implemented in INDY will continue to work, because the access methods which designers used (e.g. by date, by user, by design unit) are all amenable to indexing.

9.2 Conclusions

The work in this dissertation contributes to research in collaborative design. The asynchronous time frame has much to offer collaborating designers. Especially when the computational resources are used as a medium which supports both designing and talking about designing.

Embedded communication and *embedded history* serve to reinforce each other. Having notes near the design units to which they refer helps users interpret the design. An artifact history clarifies communication by allowing users to restore the context in which a note was first penned. These two components provide immediate benefit to collaborating designers. In addition, they provide long-term benefits to designers who are responsible for aspects of the design which were done long before they joined the project team.

This research suggests two areas for future work. First is further study on how to fully embed communication in artifacts. The current implementation has shown that though there advantages which users value, new problems are introduced which merit study.

The second area suggested for future work is an analysis of the relationship between visual representations and a design history. It appears that design history provides great benefit for users when the representation of the design is visual.

References

[Anderson 85]
J.R. Anderson, *Cognitive Psychology and Its Implications (2nd Edition)*, W.H. Freeman and Co., New York, 1985.

[Balachandran, Rosenman, Gero 91]
M. Balachandran, M.A. Rosenman, J.S. Gero, *A knowledge-based approach to the automatic verification of designs from CAD databases*, in J. Gero (ed.), *Artificial Intelligence in Design'91*, Butterworth-Heinemann Ltd, Oxford, England, 1991, pp. 757-781.

[Barnes, Bovey 86]
D.J. Barnes, J.D. Bovey, *Managing command submission in a multiple-window environment*, Software Engineering Journal, Vol. 1, No. 5, 1986, pp. 177-184.

[Bolt 84]
R.A. Bolt, *The Human Interface*, Lifetime Learning Publications, Belmont, CA, 1984.

[Borning 79]
A.H. Borning, *ThingLab -- A Constraint-Oriented Simulation Laboratory*, Technical Report SSL-79-3, Xerox Palo Alto Research Center, Palo Alto, CA, July 1979.

[Brooks 83]
R.A. Brooks, *Towards a Theory of the Comprehension of Computer Programs*, International Journal of Man-Machine Studies, Vol. 18, 1983, pp. 543-554.

[Budge 83]
B. Budge, *Pinball Construction Set (Computer Program)*, Electronic Arts, San Mateo, CA, 1983.

[Burton, Brown, Fischer 84]
R.R. Burton, J.S. Brown, G. Fischer, *Analysis of Skiing as a Success Model of Instruction: Manipulating the Learning Environment to Enhance Skill Acquisition*, in B. Rogoff, J. Lave (eds.), *Everyday Cognition: Its Development in Social Context*, Harvard University Press, Cambridge, MA - London, 1984, pp. 139-150.

[Carraher, Carraher, Schliemann 85]
T.N. Carraher, D.W. Carraher, A.D. Schliemann, *Mathematics in the streets and in the schools*, British Journal of Developmental Psychology, Vol. 3, 1985, pp. 21-29.

[Cavalier et al. 91]
T Cavalier, R Chandhok, J Morris, D Kaufer, C Neuwirth, *A visual design for collaborative work: Columns for commenting and annotation*, The 24th Hawaii International Conference on System Sciences, IEEE Computer Society, January 1991, pp. 729-738.

[Chalfonte, Fish, Kraut 91]
B.L. Chalfonte, R.S. Fish, R.E. Kraut, *Expressive Richness: A Comparison of Speech and Text as Media for Revision*, Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems, ACM, New York, 1991, pp. 21-26.

[ChenDietterichUllman 91]
A. Chen, T.G. Dietterick, D.G. Ullman, *A Computer-Based Design History Tool*, Proceedings of the 1991 NSF Design and Manufactering Systems Conference, SME (Society of Manufactering Engineers), Austin, Texas, January 1991, pp. 985-994.

[Collins, Brown 88]
A. Collins, J.S. Brown, *The Computer as a Tool for Learning Through Reflection*, in H. Mandl, A. Lesgold (eds.), *Learning Issues for Intelligent Tutoring Systems*, Springer-Verlag, New York, 1988, pp. 1-18, ch. 7.

[Comer 88]
D.E. Comer, *Internetworking with TCP/IP: Principles, Protocols, and Architecture*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[Conklin, Begeman 87]
J. Conklin, M. Begeman, *gIBIS: A Hypertext Tool for Team Design Deliberation*, Hypertext'87 Papers, University of North Carolina, Chapel Hill, NC, November 1987, pp. 247-251.

[Conklin, Begeman 88]
J. Conklin, M. Begeman, *gIBIS: A Hypertext Tool for Exploratory Policy Discussion*, Proceedings of the Conference on Computer Supported Cooperative Work, ACM, New York, 1988, pp. 140-152.

[CSTB 90]
Computer Science and Technology Board, *Scaling Up: A Research Agenda for Software Engineering*, Communications of the ACM, Vol. 33, No. 3, March 1990, pp. 281-293.

[Curtis, Krasner, Iscoe 88]
B. Curtis, H. Krasner, N. Iscoe, *A Field Study of the Software Design Process for Large Systems*, Communications of the ACM, Vol. 31, No. 11, November 1988, pp. 1268-1287.

[Dehn 93]
N. Dehn, , 1993, (Personal Communication).

[DeMarco, Lister 87]
T. DeMarco, T. Lister, *Peopleware: Productive Projects and Teams*, Dorset House Publishing, New York, 1987.

[Draper 84]
S.W. Draper, *The Nature of Expertise in UNIX*, Proceedings of INTERACT'84, IFIP Conference on Human-Computer Interaction, Elsevier Science Publishers, Amsterdam, September 1984, pp. 182-186.

[Ellis, Gibbs, Rein 91]
C.A. Ellis, S.J. Gibbs, G.L. Rein, *Groupware: Some Issues and Experiences*, Communications of the ACM, Vol. 34, No. 1, 1991, pp. 38-58.

[Ellise, Greer, Placeway, Zachariassen 87]
M. Ellis, K. Greer, P. Placeway, R. Zachariassen, *TCSH - C shell with filename completion and command line editing*, 1987, (UNIX Programmer's Manual (revised U of T, previous revisions from Fairchild, HP Labs., and OSU IRCC)).

[Engelbart, English 68]
D.C. Engelbart, W.K. English, *A Research Center for Augmenting Human Intellect*, Proceedings of the AFIPS Fall Joint Computer Conference, The Thompson Book Company, Washington, D.C., 1968, pp. 395-410.

[Fischer 87]
G. Fischer, *Making Computers more Useful and more Usable*, Proceedings of the 2nd International Conference on Human-Computer Interaction (Honolulu, Hawaii), Elsevier Science Publishers, New York, August 1987, pp. 97-104.

[Fischer 90]
G. Fischer, *Communications Requirements for Cooperative Problem Solving Systems*, The International Journal of Information Systems (Special Issue on Knowledge Engineering), Vol. 15, No. 1, 1990, pp. 21-36.

[Fischer 92a]
G. Fischer, *Domain-Oriented Design Environments*, Proceedings of the 7th Annual Knowledge-Based Software Engineering (KBSE-92) Conference (McLean, VA), IEEE Computer Society Press, Los Alamitos, CA, September 1992, pp. 204-213.

[Fischer 92b]
G. Fischer, *Shared Knowledge in Cooperative Problem-Solving Systems - Integrating Adaptive and Adaptable Systems*, Proceedings of 3rd International Workshop on User Modeling (UM'92), E. Andre et al. (eds.), The German Research Center for Artificial Intelligence, Dagstuhl, Germany, August 1992, pp. 148-161.

[Fischer et al. 90]
G. Fischer, A.C. Lemke, T. Mastaglio, A. Morch, *Using Critics to Empower Users*, Human Factors in Computing Systems, CHI'90 Conference Proceedings (Seattle, WA), ACM, New York, April 1990, pp. 337-347.

[Fischer et al. 91a]
G. Fischer, A.C. Lemke, R. McCall, A. Morch, *Making Argumentation Serve Design*, Human Computer Interaction, Vol. 6, No. 3-4, 1991, pp. 393-419.

[Fischer et al. 91b]
G. Fischer, A.C. Lemke, T. Mastaglio, A. Morch, *The Role of Critiquing in Cooperative Problem Solving*, ACM Transactions on Information Systems, Vol. 9, No. 2, 1991, pp. 123-151.

[Fischer, Girgensohn 90]
G. Fischer, A. Girgensohn, *End-User Modifiability in Design Environments*, Human Factors in Computing Systems, CHI'90 Conference Proceedings (Seattle, WA), ACM, New York, April 1990, pp. 183-191.

[Fischer, Lemke 88]
G. Fischer, A.C. Lemke, *Construction Kits and Design Environments: Steps Toward Human Problem-Domain Communication*, Human-Computer Interaction, Vol. 3, No. 3, 1988, pp. 179-222.

[Fischer, McCall, Morch 89a]
G. Fischer, R. McCall, A. Morch, *Design Environments for Constructive and Argumentative Design*, Human Factors in Computing Systems, CHI'89 Conference Proceedings (Austin, TX), ACM, New York, May 1989, pp. 269-275.

[Fischer, McCall, Morch 89b]
G. Fischer, R. McCall, A. Morch, *JANUS: Integrating Hypertext with a Knowledge-Based Design Environment*, Proceedings of Hypertext'89 (Pittsburgh, PA), ACM, New York, November 1989, pp. 105-117.

[Fischer, Nakakoji 91]
G. Fischer, K. Nakakoji, *Empowering Designers with Integrated Design Environments*, in J. Gero (ed.), *Artificial Intelligence in Design'91*, Butterworth-Heinemann Ltd, Oxford, England, 1991, pp. 191-209.

[Fischer, Nakakoji 92]
G. Fischer, K. Nakakoji, *Beyond the Macho Approach of Artificial Intelligence: Empower Human Designers - Do Not Replace Them*, Knowledge-Based Systems Journal, Vol. 5, No. 1, 1992, pp. 15-30.

[Fischer, Rathke 88]
G. Fischer, C. Rathke, *Knowledge-Based Spreadsheet Systems*, Proceedings of AAAI-88, Seventh National Conference on Artificial Intelligence (St. Paul, MN), Morgan Kaufmann Publishers, San Mateo, CA, August 1988, pp. 802-807.

[Fischer, Reeves 92]
G. Fischer, B.N. Reeves, *Beyond Intelligent Interfaces: Exploring, Analyzing and Creating Success Models of Cooperative Problem Solving*, Applied Intelligence, Special Issue Intelligent Interfaces, Vol. 1, 1992, pp. 311-332.

[Francik, Rudman, Cooper, Levine 91]
E. Francik, S.E. Rudman, D. Cooper, S. Levine, *Putting Innovation to Work: Adoption Strategies for Multimedia Communication Systems*, CACM, Vol. 34, No. 12, December 1991.

[Gance 90]
S. Gance, *Human Problem-Domain Communication in River Basin Planning and Operations*, Unpublished Master's Thesis, University of Colorado, Boulder, December 1990.

[Girgensohn 92]
A. Girgensohn, *End-User Modifiability in Knowledge-Based Design Environments*, Ph.D. Dissertation, Department of Computer Science, University of Colorado, Boulder, CO, 1992, Also available as TechReport CU-CS-595-92.

[GreenbergWitten 88]
S. Greenberg, I.H. Witten, *How users repeat their actions on computers: Principles for design of history mechanisms*, Proceedings of ACM CHI'88 Conference on Human Factors in Computing Systems, E. Soloway, D. Frey, S.B. Sheppard (eds.), ACM, New York, 1988, pp. 171-178.

[Greif 88]
I. Greif (ed.), *Computer-Supported Cooperative Work: A Book of Readings*, Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[Grudin 88]
J. Grudin, *Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'88), ACM, New York, September 1988, pp. 85-93.

[Grudin 89]
J. Grudin, *Why groupware applications fail: Problems in design and evaluation*, Office Technology and People, Vol. 4, No. 3, 1989, pp. 245-264.

[Grudin 92]
J. Grudin, *Groupware and social dynamics: Eight challenges for developers*, CACM, 1992, (in press).

[Grudin, Poltrock 89]
J. Grudin, S.E. Poltrock, *User Interface Design in Large Corporations: Coordination and Communication Across Disciplines*, Human Factors in Computing Systems, CHI'89 Conference Proceedings (Austin, TX), ACM, New York, May 1989, pp. 197-210.

[HabrakenGross 88]
N. John Habraken, Mark D. Gross, *Concept Design Games*, Design Studies, Vol. 9, No. 3, July 1988, pp. 181.

[Hackman, Kaplan 74]
J.R. Hackman, R.E. Kaplan, *Interventions into group process: An approach to improving the effectiveness of groups*, Decision Sciences, Vol. 5, 1974, pp. 459-480.

[Halasz, Moran, Trigg 87]
F.G. Halasz, T.P. Moran, R.H. Trigg, *NoteCards in a Nutshell*, Human Factors in Computing Systems and Graphics Interface, CHI+GI'87 Conference Proceedings (Toronto, Canada), ACM, New York, April 1987, pp. 45-52.

[Hill 89]
W.C. Hill, *The Mind at AI: Horseless Carriage to Clock*, AI Magazine, Vol. 10, No. 2, Summer 1989, pp. 29-41.

[Hill et al. 92]
W.C. Hill, J.D. Hollan, D. Wroblewski, T. McCandless, *Edit Wear and Read Wear*, Human Factors in Computing Systems, CHI'92 Conference Proceedings (Monterrey, CA), ACM, May 1992, pp. 3-9.

[Hollan, Stornetta 92]
J. Hollan, S. Stornetta, *Beyond Being There*, Proceedings of ACM CHI'92 Conference on Human Factors in Computing Systems, ACM, New York, 1992, pp. 119-125.

[Hutchins 90]
E. Hutchins, *The Technology of Team Navigation*, in P. Galegher, R. Kraut, C. Egido (ed.), *Intellectual Teamwork*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1990, ch. 8.

[Johansen 88]
R. Johansen, *Groupware: Computer Support for Business Teams*, The Free Press, New York, 1988.

[Joy 87]
W.N. Joy, *An Introduction to the C Shell*, 1987, (UNIX Programmer's Manual (7th edition) 2c: Supplementary Documentation).

[Kling 91]
R. Kling, *Cooperation, Coordination and Control in Computer Supported Work*, Communications of the ACM, Vol. 34, No. 12, December 1991, pp. 83-88.

[Kuffner, Ullman 91]
T.A. Kuffner, D.G. Ullman, *The information requests of mechanical design engineers*, Design Studies, Vol. 12, No. 1, January 1991, pp. 42-50.

[Kunz, Rittel 70]
W. Kunz, H.W.J. Rittel, *Issues as Elements of Information Systems*, Working Paper 131, Center for Planning and Development Research, University of California, Berkeley, CA, 1970.

[Lai, Malone 88]
K.-Y. Lai, T.W. Malone, *Object Lens: A ''Spreadsheet'' for Cooperative Work*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'88), ACM, New York, September 1988, pp. 115-124.

[Lave 88]
J. Lave, *Cognition in Practice*, Cambridge University Press, Cambridge, UK, 1988.

[Lee 92a]
A. Lee, *Investigations into History Tools for user Support*, Unpublished Ph.D. Dissertation, University of Toronto, 1992.

[Lee 92b]
L. Lee, *The Day The Phones Stopped*, Donald I. Fine, Inc., New York, 1992.

[Lemke 89]
A.C. Lemke, *Design Environments for High-Functionality Computer Systems*, Unpublished Ph.D. Dissertation, Department of Computer Science, University of Colorado, July 1989.

[Lemke 90]
A.C. Lemke, *Cooperative Problem Solving Systems Must Have Critics*, Proceedings of the AAAI Spring Symposium Workshop on Knowledge-Based Human Computer Communication, AAAI, Menlo Park, CA, March 1990, pp. 73-75.

[Lemke, Fischer 90]
A.C. Lemke, G. Fischer, *A Cooperative Problem Solving System for User Interface Design*, Proceedings of AAAI-90, Eighth National Conference on Artificial Intelligence, AAAI Press/The MIT Press, Cambridge, MA, August 1990, pp. 479-484.

[Linxi, Habermann 88]
C. Linxi, A.N. Habermann, *A history mechanism and undo/redo/reuse support in ALOE*, Technical Report CMU-CS-86-148, CMU, Pittsburgh, PA,: Department of Computer Science, 1988.

[Malone et al. 86]
T.W. Malone, K.R. Grant, K.-Y. Lai, R. Rao, D. Rosenblitt, *Semi-Structured Messages are Surprisingly Useful for Computer-Supported Coordination*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'86), MCC, Austin, TX, December 1986, pp. 102-114.

[Malone et al. 88]
T.W. Malone, K.R. Grant, K.-Y. Lai, R. Rao, D. Rosenblitt, *Object Lens: a ''spreadsheet'' for cooperative work*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'88), ACM, New York, September 1988, pp. 115-124.

[Malone, Grant, Turbak 86]
T.W. Malone, K.R. Grant, F.A. Turbak, *The Information Lens: An Intelligent System for Information Sharing in Organizations*, Human Factors in Computing Systems, CHI'86 Conference Proceedings (Boston, MA), ACM, New York, April 1986, pp. 1-8.

[Marshall, Irish 89]
C.C. Marshall, P.M. Irish, *Guided Tours and On-Line Presentations: How Authors Make Existing Hypertext Intelligible for Readers*, Proceedings of Hypertext'89 (Pittsburgh, PA), ACM, New York, November 1989, pp. 15-26.

[McCall 91]
R. McCall, *PHI: A Conceptual Foundation for Design Hypermedia*, Design Studies, Vol. 12, No. 1, 1991, pp. 30-41.

[McMahon" 87]
M. McMahon, *A practical system for managing mixed mode user interfaces*, 1987, (working paper).

[Microsoft 88]
Microsoft Word - User's Guide, Microsoft Corporation, Bellevue, WA, 1988.

[Nakakoji 93]
K. Nakakoji, *Increasing Shared Understanding of a Design Task between Designers and Design Environments: The Role of a Specification Component*, Unpublished Ph.D. Dissertation, Department of Computer Science, University of Colorado, 1993, Also available as TechReport CU-CS-651-93.

[Nakakoji, Fischer 90]
K. Nakakoji, G. Fischer, *Catalog Explorer: Exploiting the Synergy of Integrated Design Environments*, Software Symposium'90 (Kyoto, Japan), June 1990, pp. 264-271.

[Nemeth, Snyder, Seebass 89]
E. Nemeth, G. Snyder, S. Seebass, *Unix System Administration Handbook*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[Neuwirth et al. 90]
M. Neuwirth, D. Kaufer, R. Chandhok, J. Morris, *Issues in the Design of Computer Support for Co-authoring and Commenting*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'90), ACM, New York, September 1990, pp. 183-195.

[Neuwirth, Kaufer, Chimera, Gillespie 87]
C. Neuwirth, D. Kaufer, R. Chimera, T. Gillespie, *The Notes Program: A Hypertext Application for Writing from Source Texts*, Hypertext'87 Papers, University of North Carolina, Chapel Hill, NC, November 1987, pp. 121-141.

[Norman, Draper 86]
D.A. Norman, S.W. Draper (eds.), *User Centered System Design, New Perspectives on Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

[Perlin 90]
K. Perlin, *Pad Multiscale Interface Status Report*, 1990, Demo presented at the 1990 CTCT Workshop.

[Petroski 85]
H. Petroski, *To Engineer Is Human: The Role of Failure in Successful Design*, St. Martin's Press, New York, 1985.

[Polanyi 66]
M. Polanyi, *The Tacit Dimension*, Doubleday, Garden City, NY, 1966.

[Reder 82]
L.M. Reder, *Plausability judgment versus fact retrieval: Alternative strategies for sentence verification*, Psychological Review, Vol. 89, 1982, pp. 250-280.

[Reeves, Shipman 92a]
B.N. Reeves, F. Shipman, *Supporting Communication between Designers with Artifact-Centered Evolving Information Spaces*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'92), ACM, New York, November 1992, pp. 394-401.

[Reeves, Shipman 92b]
B.N. Reeves, F. Shipman, *Making it Easy for Designers to Provide Design Rationale*, Working Notes of the AAAI 1992 Workshop on Design Rationale Capture and Use, AAAI, San Jose, CA, July 1992, pp. 227-233.

[Reid, Walker 80]
B.K. Reid, J.H. Walker, Unilogic, *SCRIBE Introductory User's Manual*, Pittsburgh, 1980.

[Rittel 72]
H.W.J. Rittel, *On the Planning Crisis: Systems Analysis of the First and Second Generations*, Bedriftsokonomen, Vol. 8, 1972, pp. 390-396.

[Rittel 84]
H.W.J. Rittel, *Second-Generation Design Methods*, in N. Cross (ed.), *Developments in Design Methodology*, John Wiley & Sons, New York, 1984, pp. 317-327.

[Schoen 83]
D.A. Schoen, *The Reflective Practitioner: How Professionals Think in Action*, Basic Books, New York, 1983.

[Schoen 87]
D.A. Schoen, *Educating the Reflective Practitioner*, Jossey-Bass Publishers, San Francisco, CA, 1987.

[Schoen 92]
D.A. Schoen, *Designing as a reflective conversation with the mateirals of a design situation*, Knowledge Based Systems, Vol. 5, No. 1, March 1992, pp. 3-14.

[Shepherd, Mayer, Kuchinsky 90]
A. Shepherd, N. Nayer, A. Kuchinsky, *Strudel - An Extensible Electronic Conversation Toolkit*, Proceedings of ACM CSCW'90 Conference on Computer-Supported Cooperative Work, Association for Computing Machinery, 1990, pp. 93-104.

[Simon 81]
H.A. Simon, *The Sciences of the Artificial*, The MIT Press, Cambridge, MA, 1981.

[Stahl 92]
G. Stahl, *Toward a Theory of Hermeneutic Software Design*, Technical Report CU-CS-589-92, Department of Computer Science, University of Colorado, Boulder, CO, March 1992.

[Stallman 81]
R.M. Stallman, *EMACS, the Extensible, Customizable, Self-Documenting Display Editor*, ACM SIGOA Newsletter, Vol. 2, No. 1/2, 1981, pp. 147-156.

[Steele 80]
G.L. Steele, *The Definition and Implementation of a Computer Programming Language Based on Constraints*, Technical Report MIT-TR 595, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1980.

[Stefik et al. 87]
M. Stefik, G. Foster, D.G. Bobrow, K. Kahn, S. Lanning, L. Suchman, *Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings*, Communications of the ACM, Vol. 30, No. 1, January 1987, pp. 32-47.

[Suchman 87]
L.A. Suchman, *Plans and Situated Actions*, Cambridge University Press, Cambridge, UK, 1987.

[Sukaviriya 90]
Sukaviriya, *Coupling A UI Framework with Automatic Generation of Context-Sensitive Animated Help*, Third Annual Symposium on User Interface Software and Technology, ACM Press, New York, October 1990, pp. 152-166.

[Tanenbaum 81]
A. Tanenbaum, *Computer Networks: Toward Distributed Processing Systems*, Prentice Hall, Englewood Cliffs, NJ, 1981.

[Teitelman, Masinter 81]
W. Teitelman, L. Masinter, *The interlisp programming environment*, Computer, Vol. , April 1981, pp. 39-50.

[Thimbleby, Anderson, Witten 90]
H. Thimbleby, S. Anderson, I.H. Witten, *Reflexive CSCW: Supporting Long-Term Personal Work*, Interacting with Computers, Vol. 2, No. 3, 1990, pp. 330-336.

[Tichy 82]
W.F. Tichy, *Design, Implementation, and Evaluation of a Revision Control System*, Proceedings of the 6th International Conference on Software Engineering, IEEE, Tokyo, Japan, September 1982, pp. .

[Trigg, Suchman, Halasz 86]
R.H. Trigg, L.A. Suchman, F.G. Halasz, *Supporting Collaboration in NoteCards*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'86), MCC, Austin, TX, December 1986, pp. 153-162.

[Ullman 91]
D.G. Ullman, *The status of design theory research in the United States*, Design Studies, Vol. 12, No. 4, October 1991, pp. 204-208.

[Wenger 87]
E. Wenger, *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann Publishers, Los Altos, CA, 1987.

[Wilde, Lewis 90]
N. Wilde, C.H. Lewis, *Spreadsheet-Based Interactive Graphics: From Prototype to Tool*, Human Factors in Computing Systems, CHI'90 Conference Proceedings (Seattle, WA), ACM, New York, April 1990, pp. 153-159.

[Winograd 88]
T. Winograd, *A Language/Action Perspective on the Design of Cooperative Work*, Human-Computer Interaction, Vol. 3, No. 1, 1988, pp. 3-30.

[Wolf, Rhyne 92]
C.G. Wolf, J.R. Rhyne, *Communication and Information Retrieval with a Pen-based Meeting Support Tool*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'92), ACM, Toronto, Canada, November 1992, pp. 322-329.

[Woods 86]
D.D. Woods, *Cognitive Technologies: The Design of Joint Human-Machine Cognitive Systems*, AI Magazine, Vol. 6, No. 4, Winter 1986, pp. 86-92.

[Yakemovic, Conklin 90]
K.C. Yakemovic, E.J. Conklin, *Report of a Development Project Use of an Issue-Based Information System*, Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'90), 1990, pp. 105-118.

I. Bridge Game Instructions

Group 1 Person A Instructions

Group 1 Person A. Please refrain from speaking during the game
Step 1: Draw 3 sketches of a bridge design, 1 for strength, 1 to scale, 1 for aesthetics, each one on a separate page. Remember which is which, but don't label the sketches.
Step 2: give the sketches to the critiquer, and wait to get them back
Step 3: rate each critique: good, neutral, or bad
Step 4: redraw the designs on new sheets of paper, to satisfy the critiques. You may not agree with any critiques, but please be like Nike and "Just Do it" anyway...

Group 1 Person B Instructions

Group 1 Person B. Please refrain from speaking during the game
Step 1: wait for the designer to sketch 3 bridge designs
Step 2: guess which sketch was intended for which criterion: strength, to scale, aesthetics
Step 3: do a written critique of each sketch according to the criterion you guessed for that sketch. Make these critiques on the transparencies that you place over the sketches.
Step 4: give your critiques back to the designer

Group 2 Person A Instructions

Group 2 Person A. Please refrain from speaking during the game
Note: do not let the critiquer see these instructions
Step 1: Use the legos to build a bridge to scale. The bridge is part of an extension to the Boulder Creek path, will cross the creek, and support typical path traffic.
Step 2: give the bridge to the critiquer, and wait to get it back
Step 3: rate the critique: good, neutral, or bad
Step 4: modify the bridge to satisfy the critique. You may not agree with the critique, but please be like Nike and "Just Do it" anyway...

Group 2 Person B Instructions

Group 2 Person B. Please refrain from speaking during the game
Note: do not let the designer see these instructions
Step 1: wait for the designer to build a lego bridge whose primary design criterion is strength. The Folsom Street bridge over the Boulder Creek is being replaced and will need to support truck traffic.
Step 2: do a written critique of the bridge with respect to the goal of strength and use post-it notes to make your critique; attach them to the bridge where appropriate
Step 3: give the critiqued bridge back to the designer

Group 3 Person A Instructions

Group 3 Person A. Please refrain from speaking during the game
Note: do not let the critiquer see these instructions
Step 1: Use the legos to build a bridge for aesthetics. Any kind of bridge you want to design is ok, just make it aesthetically pleasing.
Step 2: give the bridge to the critiquer, and wait to get it back
Step 3: rate the critique: good, neutral, or bad
Step 4: modify the bridge to satisfy the critique. You may not agree with the critique, but please be like Nike and "Just Do it" anyway...

Group 3 Person B Instructions

Group 3 Person B Please refrain from speaking during the game
Note: do not let the designer see these instructions
Step 1: wait for the designer to build a lego bridge whose primary design criterion is aesthetics
Step 2: do a written critique of the bridge with respect to the goal of aesthetics using an 8-1/2 x 11 sheet of paper
Step 3: give the critique and bridge back to the designer

Group 4 Person A Instructions

Group 4 Person A. Please refrain from speaking during the game
Note: do not let the critiquer see these instructions
Step 1: Use the legos to build a bridge for strength. The Folsom Street bridge over the Boulder Creek is being replaced and will need to support truck traffic.
Step 2: give the bridge to your partner, who may redesign it
Step 3: do a written critique of the modified bridge, using post-it notes

Group 4 Person B Instructions

Group 4 Person B. Please refrain from speaking during the game
Note: do not let the designer see these instructions
Step 1: wait for the designer to build a lego bridge whose primary design criterion is scale. The bridge is part of an extension to the Boulder Creek path, will cross the creek, and support typical path traffic.
Step 2: rebuild the bridge yourself with respect to the goal of scale
Step 3: give the rebuilt bridge back to the designer