Paper presented and accepted for publication in Proceedings of CAAD Futures '95: The Sixth International Conference on Computer-Aided Architectural Design Futures, 24-26 September 1995, Singapore.

Beyond the Shared Whiteboard: Issues in Computer Supported Collaborative Design

Wassim M. Jabi
Doctoral Program in Architecture
The University of Michigan
2000 Bonisteel Blvd.
Ann Arbor, MI 48109-2069
U.S.A.

Theodore W. Hall
Department of Architecture
The Chinese University of Hong Kong
Shatin, New Territories
Hong Kong

Abstract

This research focuses on combining the rich representations of computer-aided design systems with current collaboration technologies to support distributed design processes. Our emphasis is not on concurrent multi-user access to integrated databases, but rather on shared protocols of interaction that are independent of implementation and storage schemes. We have developed a prototype for a Synchronous Collaborative Design System (SYCODE) that enables geographically dispersed designers to share common representations even when using different hardware platforms. The limitations of the existing network infrastructure has impelled us to devise a meaningful and parsimonious representation scheme and to semantically define pending and confirmed actions.

Keywords: *Computer Supported Co-operative Work, Collaborative Design, Multi-user Synchronous CAAD, Shared Workspace, Shared Protocols of Interaction.*

1. Introduction

The traditional view of the architect as lone hero and ultimate creator of "good" architecture has been gradually abandoned, in favor of greater attention to the importance of collaboration in assuring excellence in design (Cuff, 1992). Additionally, recent advances in computer and communication technology have enabled researchers to explore a variety of computer systems that attempt to respond to this paradigm shift (Maher et. al., 1993).

Nevertheless, we find that current collaborative "groupware" technologies - such as unstructured shared whiteboards, multi-media electronic mail, desktop conferencing systems, and distributed single-user software - are inadequate for the particular needs of collaborative design processes. On the one hand, CAD systems do not support simultaneous, multi-user discussion and co-production of architectural documents. On the other hand, the limited data structures of generic groupware systems do not capture architecturally significant semantics, such as mass, material, space, and function.

To expand the capabilities of current CAD systems and to go beyond the limited expressiveness of general purpose collaboration tools, we propose a new generation of design-oriented software that combines collaboration technologies with a rich and meaningful

representation scheme. We are particularly interested in supporting the early design phases, wherein many of the most important decisions are made and collaboration is most important: client debriefing, data collection, architectural program formulation, and schematic design generation. These activities are crucial to the evolution and quality of the final design, and they are receptive to and can benefit from computer support. Furthermore, these are precisely the areas where current CAD systems are weakest.

We propose a Synchronous Collaborative Design Environment (SYCODE) that enables teamwork among geographically dispersed designers using the Internet. SYCODE is a long-term project which aims at supporting the creation of global design teams. This paper provides an overview of the theoretical foundations of SYCODE and an outline of its overall architecture, communication protocols, and data structures. In addition, it includes results from our first experimental phase, which involved the simultaneous development and testing of SYCODE clients on heterogeneous computer systems at remote sites - Hong Kong and Ann Arbor, Michigan.

2. The Client-Server Model

SYCODE is based on the Share-Kit (Jahn, 1994) which is a general-purpose toolkit that enables the construction of synchronous groupware systems. The Share-Kit uses a client-server paradigm in which computer programs, called clients, exchange messages through a central computer program called a server (Figure 1). In the case of the Share-Kit, the server offers multiple sessions that act as switching boards. Clients can exchange information when they connect to the same session within the central server. This can be achieved through a communication interface module that translates the data being shared between its parent client and the server. Beyond this interface, the client's specific implementation is independent from that of the server and other clients.

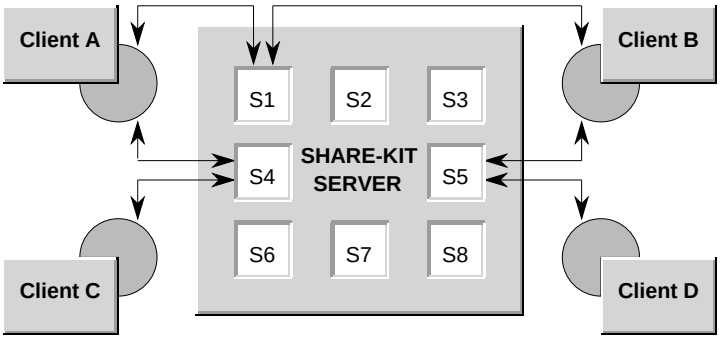

Figure 1. The Share-Kit's Client-Server Design.

3. Issues in Computer-Supported Collaborative Design

This research project grew out of a case-study that analyzed the role of artifacts in collaborative design (Jabi, 1995). It concluded with the finding that, in architectural design, supporting focused tasks is more beneficial than supporting general-purpose ones and that artifacts, to a great extent, act as facilitators of collaboration. A disadvantage of collaborative whiteboard systems that represent the shared workspace as a video image or a digital bitmap is that they do not easily allow for the interpretation of the shared data by the connected computers

(Tang and Minneman, 1991). Additionally, researchers are finding that these general-purpose multimedia applications are hindering richer ways of collaboration:

> Collaborative multimedia applications have been somewhat lacking in imagination, focusing on "talking heads" video as a way to create telepresence, or on data retrieval for simple information "foraging" and sharing tasks … We argue that collaborative multimedia technologies should be used … as means of providing richer, deeper ways to collaboratively compose shared artifacts such as documents, movies, data visualizations, simulations, designs such as architectural drawings, bulletin boards, libraries, and animations. We should also create ways to collaboratively analyze data within these artifacts.(Nardi, 1994).

Many of the collaborative tools that exist today can only support the hardware platform they were developed on (Watabe et. al., 1990). If we are to succeed in connecting users from various backgrounds together, we must be able to accommodate the extensive variety in hardware, configuration, display systems and operating systems that exists. To illustrate the feasibility of achieving this goal, SYCODE client applications were developed independently, based on a verbal description of protocols, with minimal sharing of actual source code. The first site - in Hong Kong- used a Silicon Graphics computer, the C programming language, and the X Window System, while the second site - in Ann Arbor, Michigan - used a NeXT workstation, a combination of C and Objective-C programming languages, and Display Postscript. Though their user interfaces and implementation details are different, these prototypes allow multiple users to share a virtual design space - both within and between the remote sites - in which to create and manipulate architectural elements.

Parsimony is another important goal we sought in our design of the SYCODE architecture. There are at least two layers of issues in implementing domain-specific software for computer-supported collaborative work. The higher layer, concerning the semantics of architectural design teamwork, is rightly the principal focus of this research effort. Nevertheless, the lower layer, concerning the technical details of long-distance network communications, rears its ugly head and imposes limits on what can actually be accomplished with the existing infrastructure.

Five universities in Hong Kong rely on a single satellite dish for all of their Internet communications with North America. From this dish, the signal travels over 22,000 miles to a satellite in geosynchronous orbit, thence another 22,000 miles to a dish in California. The data rate over this path, as reported by programs such as "ping", "ftp", and "Netscape", rarely exceeds 100 bytes per second, and often falls below 50. This is on the order of 1000 times slower than the typical rate for local network communications at either end. Furthermore, the percentage packet loss as reported by "ping" is generally in the double digits.

If one wishes to move beyond theorizing about computer-supported collaborative work and discuss actual implementations, then one must come to terms with the limitations imposed by the network. We anticipate the day when the tenuous satellite link is replaced by a solid fiber-optic connection, but in the meantime … The low transmission rate, compounded by the loss and consequent retransmission of a large percentage of packets, results in time delays that are significant enough to become issues at the higher semantic layer of the design system (as well as in the details of how to program the main event loop).

For example, when a client changes an element of the model, when is the change considered to be officially "done": Immediately? After he/she receives a round-trip confirmation from the server? Or after all clients have received the message from the server? How can a client know that all other clients have received the message, except by requesting a confirmation from each -

thus multiplying the communication traffic and compounding the problem?

If each client considers his/her actions to have immediate effect, then how are discrepancies in other clients' simultaneous states of the model resolved? Where does the "official" state reside, and how is it restored and redistributed in the event of a network failure?

As a developer or proposer of a collaborative design system, one may adopt the philosophy that these are transient technical problems that can be resolved with improved network infrastructure and robust software. However, the current weaknesses of long-distance network computing suggest an alternative philosophy that may lead to a richer system design. If one adopts the view that discrepancies are inherent in human interaction, then the goal is not to eliminate them, but to manage them. Technical glitches in the computing infrastructure are only one source of discrepancy.

Our aim in designing SYCODE is not to automate design, but to support collaboration between human designers, working in proximate or remote locations, simultaneously or in "shifts". Thus, we accept that discrepancy from a variety of sources is inevitable, and must be managed at the higher semantic layer of the system as well as in the details of programming event loops and Interment socket communications.

4. An Overview of SYCODE

We are still in the process of specifying the overall architecture of SYCODE, thus, this section includes our provisional thoughts. These design decisions are by no means finalized and will most probably change in the future. At the conference presentation, we will present the latest modifications and seek the audience's feedback.

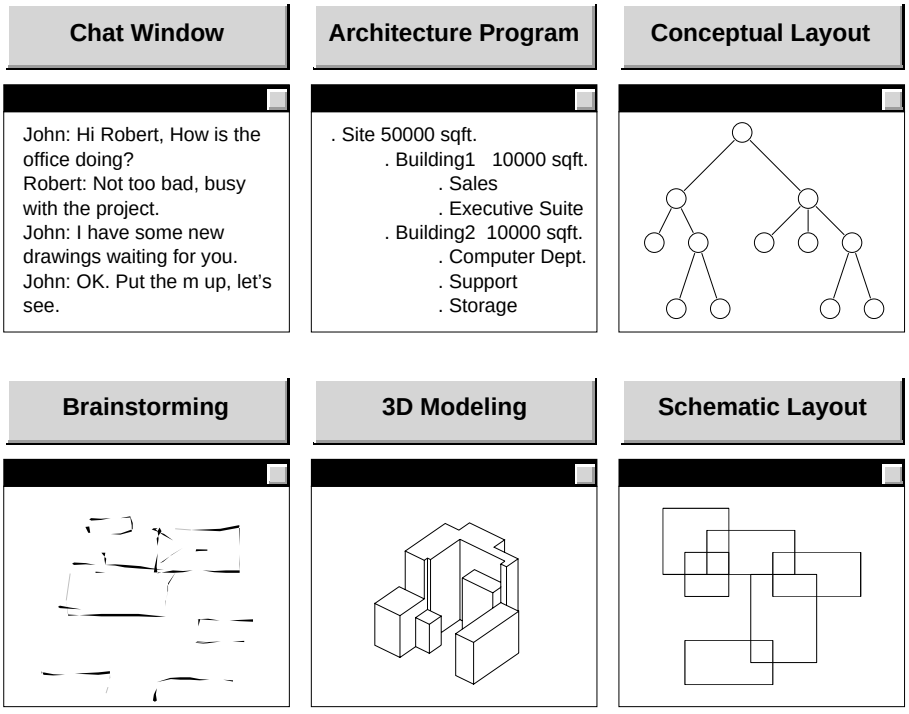

Figure 2. Multiple Views in a SYCODE Interface.

SYCODE's goal is to support the early phases of architectural design that include: design argumentation, defining spatial and non-spatial requirements, and incremental generation of schematic design solutions. Hence, a SYCODE interface offers multiple windows each representing one aspect of the design project (Figure 2). Furthermore, the underlying representation scheme used consists of agents and artifacts. Agents are representations of either human participants or software modules that possess certain skills while artifacts are the various documents and pieces of information being shared.

4.1 Agents

SYCODE defines several types of agents (Figure 3): 1) One agent is a Group Overall Director (G.O.D.) that has ultimate authority to perform any action in SYCODE. The G.O.D.'s responsibilities include defining protocols for interaction, assembling design teams, and initiating collaborative projects. 2) Each design team member is also an agent possessing certain rights with regard to artifacts. 3) Artificial agents consist of computer programs that interact with other entities in SYCODE. One such artificial agent, called the manager, is crucial for maintaining consistency. The manager remains active at all times, contains the current "official" status of the project, continuously saves it to disk to prevent data loss in case of network failure, and maintains other book-keeping information and preferences. The G.O.D. interacts with the manager to set all preferences and provide the needed information to manage the project at hand.

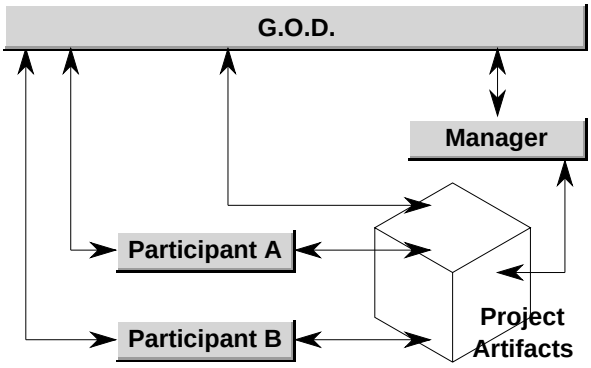

Figure 3. The Various Agents in SYCODE.

4.2 Artifacts

Artifacts in SYCODE are organized in a hierarchic fashion (Figure 4). A prototypical artifact includes links to component artifacts within it and maintains an access control list of agents that specifies their ownership status and rights for reading and modifying it. At the top of the hierarchy is a project artifact that corresponds closely to its architectural counterpart as found in professional practice. Being an artifact, it inherits the ability to include component artifacts and maintain an access control list. In addition, it maintains a record of its evolution through a directed graph representing states of the project through time. A state artifact can be thought of as a snapshot of the project at a certain point in time. Thus, SYCODE always represents the current status of a project through a current state artifact. A state is composed mainly of workspaces in which to create entities. Currently, the design of SYCODE requires one public shared workspace where the current state resides. In addition, each participant has a private workspace, access to which he/she may grant or deny to any subset of other participants. For

entities in private workspaces to become a part of the current state, they must eventually be incorporated in the public workspace. The entities in a workspace may be either declarative or procedural. Examples of declarative entities include spatial and non-spatial requirements, primitive shapes, instantiations of shapes, and architectural spaces. Examples of procedural entities include proposals for action and calls for votes.

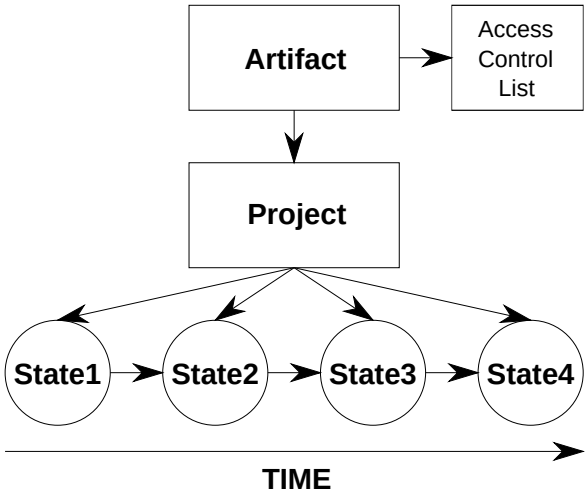

Figure 4. The Hierarchy of Artifacts in SYCODE.

As shown in figure 5, one way to maintain consistency while accommodating the often simultaneous design states is to devise a system wherein starting from an initial agreed-upon state, several individual and possibly discrepant states can be created that, nevertheless, converge at a later date on what we call a milestone state to which everyone concerned agrees. This process of convergence is achieved through either an authoritative action (such as one from the G.O.D.) or voting. To maintain the integrity and chronology of events, we have provisionally decided that participants cannot modify the public workspace unless connected to the session manager. When not connected, due to network failure or otherwise, the participant can only modify his/her private workspace. We are currently working on the implementation details of a suitable voting mechanism and will report on it at the conference presentation of this paper.

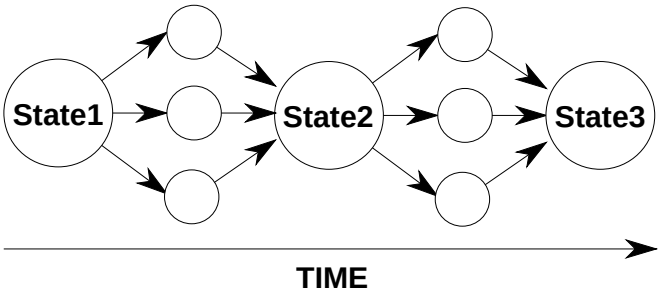

Figure 5. Divergence and Convergence of Design States.

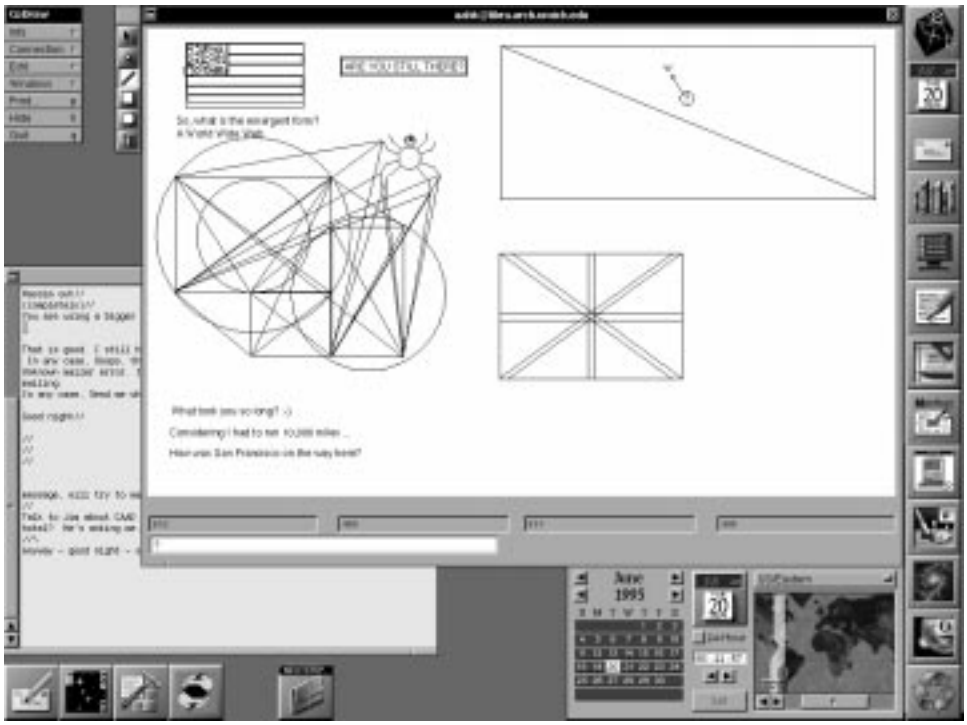

Figure 6. A SYCODE client running on a NeXT workstation.

5. Implementation

The first phase of this project included the installation of the Share-Kit servers at the two sites – Ann Arbor and Hong Kong – and the definition of an initial protocol for communication of shared artifacts. Two SYCODE client prototypes were then developed independently on different hardware platforms. The SYCODE client starts by requesting the address of the machine where the server is running, the session name to connect to, and an optional alias for the participant to be identified with. Once a connection is made, the participant can create entities and add them to the shared workspace (Figure 6). These entities, however, are not merely graphical. They behave as complete objects that can be communicated with, asked to perform certain actions such as drawing themselves; and queried about ownership, status, assigned unique identifier and other task-specific attributes.

A second client, connecting to an existing session, automatically receives a complete and synchronized copy of the current state, after which actions taken by any of the connected clients are broadcast to each of them (Figure 7). It is important to note here, that what is being transmitted are not display system events, but requests to create and modify objects. Consistency in the chronology of actions, as seen by each client, is maintained by routing all actions through the server. Thus, client actions that modify the current state do not take effect until distributed by the server and echoed back to the originator.

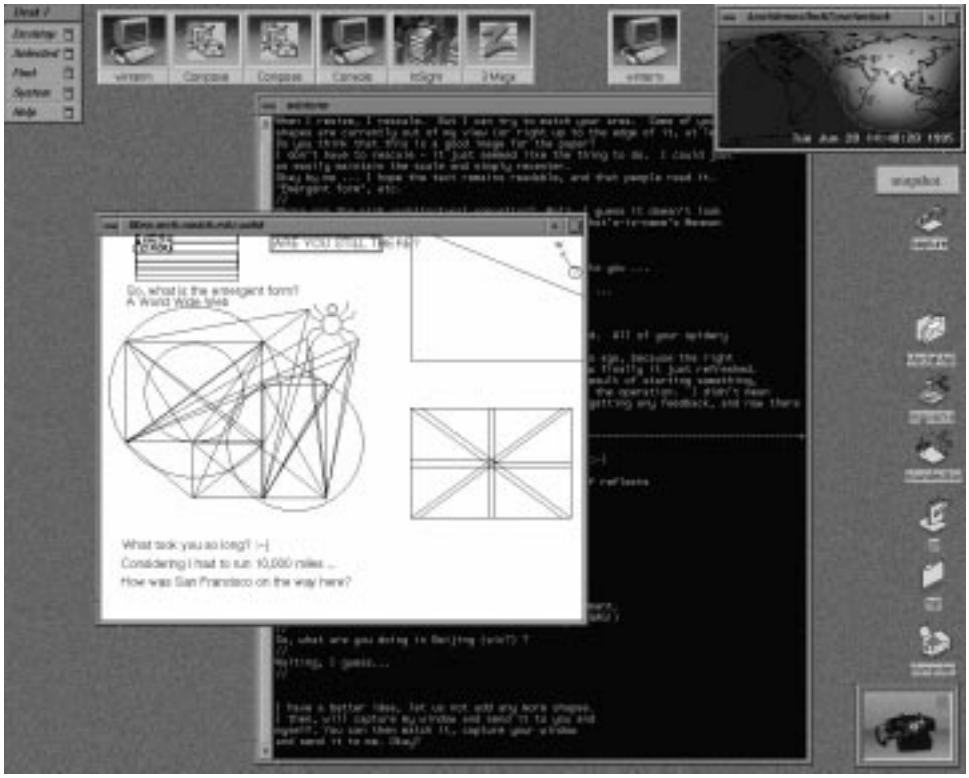

Figure 7. A SYCODE client running on a Silicon Graphics workstation.

6. Conclusion

The reality of the existing network infrastructure has dictated many of the design decisions we had to make. Parsimony in the exchange of data became a major concern for the design of SYCODE. Conducting an experiment between two vastly remote sites has informed us of the realities of time delays and taught us the need to represent pending as well as confirmed actions. Furthermore, display-specific attributes such as font style and size, window size and current scrolling position seem to be needed if tight-coupling of artifact representation is desired. Nevertheless, our experiments indicate that collaboration in architectural design can proceed effectively through the sharing of simple and concise messages that create and modify artifacts. The urgent need is to exchange and interpret artifacts rather than to share visual images of each other or shallow representations of drawings. It may well be beneficial to include video conferencing and shared whiteboards in a comprehensive collaborative environment, but we believe that sharing a common and meaningful language of design through artifacts will better serve the domain-specific requirements of architects.

References

Bly, S.A. *A Use of Drawing Surfaces in Different Collaborative Settings*, Proceedings of the Conference on Computer-Supported Cooperative Work. (New York: ACM, 1988); p. 250-256.

Cuff, D. *Architecture: The Story of Practice.* (Cambridge: MIT Press, 1992).

Jabi, W. *An Outline of the Requirements for a Computer-Supported Collaborative Design System*, Open House International (forthcoming in 1995).

Maher, L.M., Gero, J.S., Saad, M. *Synchronous Support and Emergence in Collaborative CAAD*. in CAAD Futures '93: Proceedings of the Fifth International Conference on Computer-Aided Architectural Design Futures. (New York: North-Holland, 1993); p. 455-470.

Nardi, B. (Organizer). *Collaborative Multimedia: Getting Beyond the Obvious*, Proceedings of the Second ACM Conference on Multimedia (New York: ACM, 1994); p. 119-120.

Tang, J.C. and Minneman, S.L. *VideoWhiteboard: Video Shadows to Support Remote Collaboration*, Proceedings of CHI'91. (New York: ACM, 1991); p. 315-322.

Watabe, K. et. al. *Distributed Multiparty Desktop Conferencing System: MERMAID*, Proceedings of the Conference on Computer-Supported Cooperative Work. (New York: ACM, 1990); p. 27-38.