Jabi, Wassim and Theodore Hall. "The Role of Computers in Synchronous Collaborative Design." Proceedings of the 14th International Congress on Cybernetics. Namur, Belgium: International Association for Cybernetics, 1995, pp. 71-76.

The Role of Computers in Synchronous Collaborative Design

Wassim M. Jabi, *The University of Michigan*
Theodore W. Hall, *Chinese University of Hong Kong*

Abstract

In this paper we discuss the role of computers in supporting real-time synchronous design among geographically dispersed team members using the global network of computers known as the *Internet*. To enable efficient and functional synchronous design activity, we advocate a new generation of design-oriented software that combines collaboration technologies with a meaningful and parsimonious 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. 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. As a general theoretical direction, our emphasis is not on integrated databases, but rather on shared protocols of interaction that are independent of implementation and storage schemes. Our first experimental phase involved the simultaneous development and testing of prototypes for a Synchronous Collaborative Design Environment (SYCODE) on heterogeneous computer systems at two geographically dispersed sites. The applications were developed independently, based on a verbal description of protocols, with minimal sharing of actual source code. 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.

Keywords

Architecture, Synchronous Design, Collaboration, Computers, CAD, Internet.

Introduction

In her book (Cuff, 1992) Dana Cuff notes: "good buildings are not designed by committee. Instead, excellent projects are designed by a few leaders, who, working together, are able to move the project along and coordinate the group of contributors." She later also notes that "one of the first items of propaganda that must be challenged is the primacy of the independent practitioner working with relative autonomy. This myth É is not unique to architecture but is embedded in the cultural system of all professions." Cuff identifies, through the above comments, an emerging attitude within the field of architecture: the practice of architecture does not fit a collaborative ideal where all contributors play an equal role, but excellence in design does depend on a collaborative process that involves many participants, each playing a different role.

In this paper we discuss the role of computers in supporting real-time synchronous design among geographically dispersed team members using the global network of computers known as the *Internet*. In particular, we describe a Synchronous Collaborative Design Environment (SYCODE) that supports the early phases of design within the field of architecture. For its communication infrastructure, SYCODE makes use of a general-purpose toolkit, called the Share-Kit, for developing groupware using the C programming language (Jahn, 1994). The Share-Kit allows the construction of collaborative environments that can communicate across different hardware platforms and graphics display systems. While the Share-Kit provides the ability to build collaborative systems, it does not specify their functionality. SYCODE, on the other hand, provides a domain-specific framework that specifies: 1) a design protocol based on shared definitions of data structures and methods, 2) a social hierarchy based on access rights, and 3) temporal awareness based on design states. Using only a verbal description of protocols, with minimal sharing of actual source code, applications were developed independently at the University of Michigan and the Chinese University of Hong Kong that allow multiple users to share a virtual design space - both within and between the remote sites - in which to create and manipulate architectural elements.

Requirements

The field of Computer-Supported Collaborative Work (CSCW) has produced numerous systems that allow the sharing of virtual workspaces (Bly, 1988), (Tang and Minneman, 1991). These general-purpose systems typically allow multiple user to view and annotate a shared virtual whiteboard represented either as a video image or a digital bitmap. The main disadvantage of these systems is that they do not easily allow computers to interpret the artifacts being shared – a feature found to be useful for collaborators (Olson and Olson, 1991). That is, although these methods seem to allow the complete sharing of a virtual workspace, they lack the parsimony and structure to make them useful for synchronous collaborative design. Therefore, the primary requirement for SYCODE was to provide an environment that enables the sharing of semantically relevant data structures while maintaining a low communication overhead.

Several collaborative systems can facilitate communication only among identical hardware platforms, operating systems, or graphic display systems and some even require specially equipped rooms (Tatar et al, 1991). However, realizing the diversity in computer systems that exists, an early requirement for SYCODE was to isolate it, to the largest degree possible, from any particular hardware platform or graphic display system. At minimum, we required the ability to communicate between two different hardware platforms to illustrate the potential for a platform-independent implementation.

The Share-Kit

The Share-Kit is a general-purpose toolkit that facilitates the construction of synchronous groupware. It supports the development of new systems as well as the conversion of existing single-user systems to groupware applications. As shown in figure 1, the Share-Kit uses a *client-server* paradigm that allows computer programs, called *clients*, to send and receive

information by connecting to the same *session* offered by a central computer program called *the server.* In figure 1, for example, clients A and B are communicating through session 1 while clients C and D are communicating through session 2. The different shapes for the clients illustrated below indicates the fact that they can differ in implementation details, exist on heterogeneous platforms, and react differently to the same message. However, they have in common a communication module that embodies a finite set of message protocols.

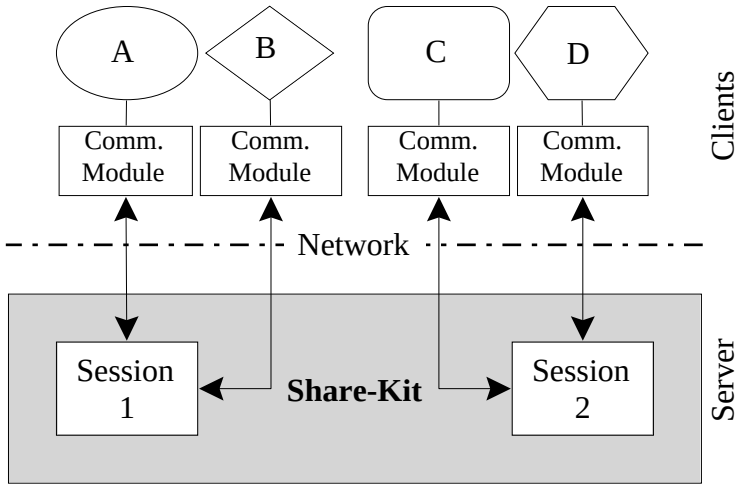

Figure 1: The Share-Kit's client-server architecture.

The Share-Kit's basic mechanism depends on a *shared function* and a *shared data representation*. A shared function is written in the C programming language and declared to the Share-Kit, along with relevant C data structures, to establish a common interface or protocol for collaboration. Using this protocol definition, the Share-Kit generates a communication module that broadcasts messages, through a central server, to all connected clients whenever a shared function is called from any of the clients.

Additionally, The Share-Kit offers a suite of management functions that enable synchronous collaboration, state updating, and state consistency. For example, when a client calls a shared function, the Share-Kit server distributes the message in real-time (aside from any network delays) to all connected clients. Moreover, when a new client connects to a session already in progress, the server automatically instructs one of the already connected clients to update the new comer with the current state. It is important to note, however, that the server itself does not store a description of the current state. Finally, while updating a new comer, the server defers and buffers all requests from other clients to insure complete consistency of the current state among all connected clients. Other features include: an optional token mechanism that grants exclusive broadcast rights to the client in possession of the token; and the ability to query the number of connected users.

SYCODE

SYCODE is a long-term project to construct a Synchronous Collaborative Design Environment based on a case-study that analyzed the role of artifacts in collaborative design and yielded a list of requirements for a computer-supported collaborative design system

(Jabi, 1995). SYCODE's goal is to support the early phases of architectural design by facilitating the dynamic social processes of design argumentation, defining spatial and non-spatial requirements, and incrementally generating schematic design solutions. To serve this goal, SYCODE's underlying architecture represents concepts and entities found in professional practice. The representation scheme is divided between *agents* and *artifacts*. Agents consist of representations of either human participants or software modules that possess certain skills. Artifacts, on the other hand, represent the virtual counterparts of objects that architects and others deal with in their profession. Given this distinction, SYCODE can then represent the interactions and relationships among agents, among artifacts, and among agents and artifacts.

SYCODE defines several types of agents: 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*, remains active at all times, contains the current status of the project, and maintains 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.

Artifacts in SYCODE are organized in a hierarchic fashion. At the top of the hierarchy is a *project* artifact that corresponds closely to its architectural counterpart as found in professional practice. It includes links to component artifacts and to the various agents that are involved in creating and modifying them. 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 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 the 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.

As illustrated in figure 2, we envision collaborative design activity as an evolving, but iterative process of divergence from an initial agreed-upon state to multiple concurrent states and their subsequent convergence to a single state through a process of voting or by the actions of agents with authority such as the G.O.D.

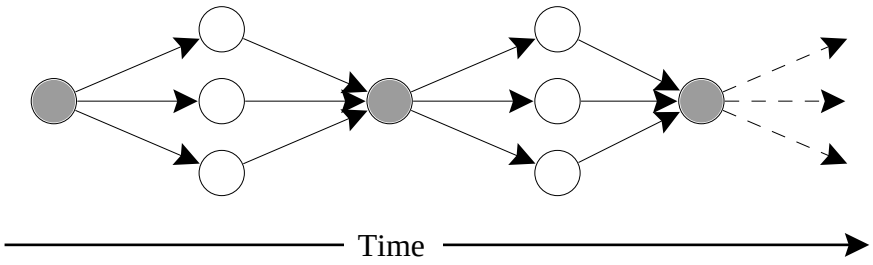

Figure 2: Progression of design through divergence and convergence.

Implementation

The initial experimentation phase involved the installation and testing of the Share-Kit on different hardware platforms at the University of Michigan at Ann Arbor and the Chinese University of Hong Kong, the definition of an initial protocol for interaction, and the independent development of two software modules that can communicate by sharing a common protocol. As illustrated in figure 3, the user is presented with a dialog box that specifies the server machine name, the session name, and an alias he/she wishes to be identified with. After establishing a connection to the server, the user can create different entities in the graphic window. Figure 4 illustrates how a second client, connecting to an existing session, 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. 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.

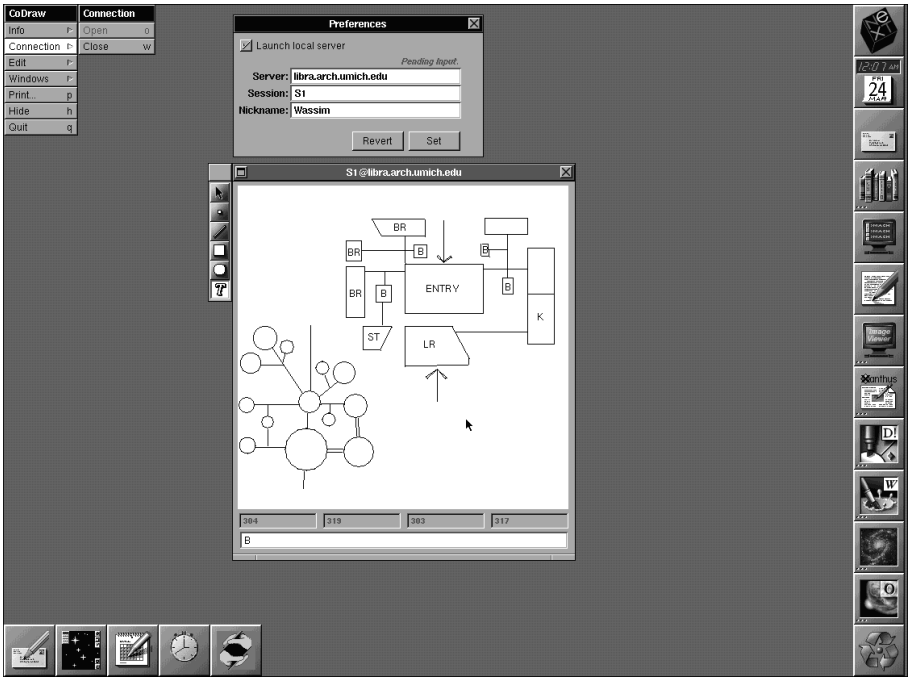

Figure 3: Screen shot of prototype (single client).

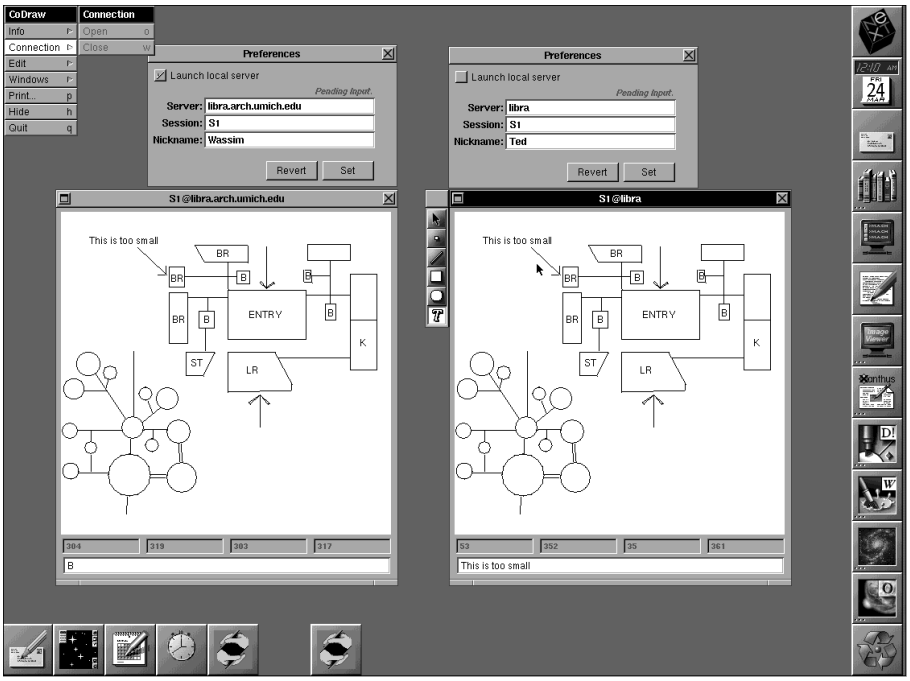

Figure 4: Screen shot of prototype (two clients).

In our informal tests, the transmission of an action between Ann Arbor and Hong Kong consumed an average of three to four seconds for a one-way trip (six to eight seconds for a round-trip). A client having a faster network path to the server experienced almost no delay between initiating an action and getting feedback. On the other hand, a client having a slower connection to the server experienced a time lag during which its action produced no feedback until the server distributed the request to all clients and finally echoed it back. To improve local user interaction, the client was modified to provide immediate graphic indication of pending actions. We expect problems with time lag to diminish as network capacity increases.

Conclusion

Starting from the premise that collaboration is necessary for excellence in design, this paper investigated the role of computers in supporting synchronous collaborative processes among geographically dispersed design team members. To enable synchronous design, computers must go beyond general-purpose shared whiteboards and video conferencing toward a representation scheme that significantly maps the semantics of the task at hand. Furthermore, time delays must be overcome by maintaining parsimony in communication and data representation. Our experience with drawing among vastly remote sites uncovered the need to indicate *pending* actions as well as *confirmed* ones.

Most importantly, however, computer support for synchronous collaborative design will succeed only when the emphasis is broadened from implementing integrated databases to inventing a consistent vocabulary for the sharing of artifacts, agents, and actions independent of storage schemes and implementation details.

References

Bly, S.A. (1988); A Use of Drawing Surfaces in Different Collaborative Settings; Proc. CSCW 88, ACM Press (pp. 250-256).

Cuff, D. (1992); Architecture: The Story of Practice; MIT Press (pp. 242-251)

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

Jahn, P. (1994); Share-Kit User Manual; Department of Computer Science of the Technical University of Berlin, Germany.

Olson, G.M. and J.S. Olson (1991); User-Centered Design of Collaboration Technology; Journal of Organizational Computing; VOL. 1, (pp. 61-83)

Tang, J.C. and S.L. Minneman (1991); VideoWhiteboard: Video Shadows To Support Remote Collaboration; Proc. CHI'91 (eds. S. Robertson, G.M. Olson, J.S. Olson), ACM Press (pp. 315-322)

Tatar, D.G., Foster, G. and D.G. Bobrow (1991); Design for Conversation: Lessons from Cognoter; International Journal of Man-Machine Studies; VOL. 34, NO. 2 (pp. 185-209)