CoLab, Tools for Computer-Based Cooperation
A Proposed Research Program

Gregg Foster

Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley 94720

ABSTRACT

CoLab is a laboratory to experiment with new forms of computer-assisted collaboration. We argue that current tools for supporting meetings are antique. We propose experiments using modern computational and display technologies to build tools for better support of meetings and cooperative problem solving. Research objectives are outlined, specifically: the goals of the project and our approach to computer-based support for cooperative problem solving and the experimental basis for CoLab. We take a quick tour of the Colab meeting lab being constructed. We outline an example tool and discuss some possible future tools. Previous software systems for supporting group work and some past efforts at structuring group problem solving are described. We present dimensions of tool design and some experiments under consideration. The basic architecture and the software primitives for group use of computers are presented. We discuss the current status of the CoLab project and our immediate plans. We plan to use CoLab to explore the use of computer software and advanced display devices to enhance and extend group problem solving activity. It will also be used as a laboratory to investigate appropriate structures for computer-based meetings.

1. Introduction

We live in a complicated world with complex problems that need to be solved. Other people and computers are probably our two most useful problem solving aids. We often need help so we call meetings to exchange information and approach problems. We also spend a lot of time interacting with computers. But when we want to get people together to intensively explore a problem we leave our computers behind and use antique technological support. Blackboards, paper and pencil, and even slide projectors are 19th century (or earlier) technology. Meetings relying on these old technologies are unnecessarily handicapped. Arguments are often lost or forgotten. Overlapping group participation is difficult. Strong personalities tend to dominate. Exploration structures inappropriate to the issue at hand are pressed into service. The well known "committee failure" problem shows us that committees tend to explore issues poorly and are normally ineffective at solving problems.

We propose the use of modern computational and display technologies to build tools to better support meetings and cooperative problem solving.* Although networks connect computers and enable electronic mail and sharing of facilities, computer systems usually aren't designed for group activities. When we want to use our computers for demonstrations, several people must gather around a display designed for a single person. When records of a collaborative session must be entered into computer systems as a separate step secondary ideas, arguments, and random notes are often lost, misrepresented, or forgotten.

Recent technological advances (e.g. Ethernet, mice, EvalServer†, bitmapped displays and windows)[1,2,3,4] have made collaborative software tools and new techniques of group problem solving possible. High quality displays and speedy communication between machines make tools for mediating and enhancing group activities feasible and an interesting topic for research.

The rest of this proposal presents *CoLab,* a laboratory to investigate new forms of computer-based collaboration. Section 2 outlines the overall research objectives and the planned approach. Section 3 presents previous work in computer-based cooperation and meetings. Section 4 mentions the dimensions for study, some experiments, and evaluation criteria. Section 5 presents the CoLab architecture in more detail and discusses its current status.

2. Research Program

The research plan has four parts.

1. Design and build software primitives for multi-machine display, data synchronization, and communication.
2. Explore software tools and the necessary tool dimensions for various forms of cooperative problem solving, so a tool (or suite of tools) can be built.
3. Experiment with these tools, their parameters and the structures and techniques they enforce and encourage.
4. Redesign the tool(s) using evaluations of the experimental tool(s).

* The terms "Meeting", "Group Problem Solving, and "Cooperative Problem Solving" are used in this proposal more or less interchangeably. Technically, *meeting* refers to a group of people gathered in the same place for any sort of information exchange. *Cooperative problem solving* or *group problem solving* refers to people working together to deal with a problem (though not necessarily in the same place or at the same time).

† EvalServer is a Lisp version of Remote Procedure Call. It evaluates S-expressions on remote machines, using the remote environment, and returns the value (if desired).

2.1. Goals

The following are the primary goals of this work:

- To see how new technology can enable new kinds of meetings and enhance cooperative problem solving.
- To discover the dimensions of real-time tools critical for their utility in meetings.
- To determine the appropriate software primitives to provide a flexible base for support of a large class of interactive real-time group tools.
- To find the appropriate structures and user interfaces for different classes of meetings.

Secondary goals include determining the classes and techniques of cooperative problem solving suitable for computer application, capturing the methodology of meetings and group problem solving, testing the effects of LiveBoard* technology on meetings, and developing a terminology for talking about computer-based meetings.

2.2. The Meeting Lab

The initial experimental environment is the CoLab meeting lab at Xerox PARC. The meeting lab is small conference room equipped with specially designed desks each holding a Xerox 1100 display (with bitmapped screen, keyboard, and optical mouse). The desks are designed to allow the displays to be raised and lowered for maintaining eye contact among participants. The desks can be arranged in a variety of formations. Computer displays in the meeting lab can be slaved to one another if desired. The computational backing is Xerox D-machines (especially Dorados) running LOOPS (Lisp Object Oriented Programming System)[5] and InterLisp. The machines are connected by Ethernet and run EvalServer (an InterLisp utility that allows Lisp S-expressions to be evaluated in a remote environment). Under development is a *LiveBoard,* a touch sensitive whiteboard coupled to a projected display and *Electronic Chalk,* the "mouse" for the LiveBoard. At a later time we expect to add video cameras and microphones for analysis of CoLab use.

* See "The Meeting Lab".

2.3. Hypotheses

The following are the major working hypotheses:

- Computer-based *Intellectual Teamwork* tools will enable new kinds of meetings and new meeting structures.
- Structured tools can elicit better response from the people involved in cooperative problem solving.
- New computational display/input-output technologies enable formerly difficult forms of group interaction such as simultaneous input, relaxed WYSIWIS (What You See Is What I See), and anonymity.
- Software tools can express appropriate group problem solving techniques and effective structuring.

2.4. Example Tool: Cognoter

Tool dimensions have not been explored sufficiently to commit to a design for a central tool. However, Cognoter is presented here in more detail to give the feel of what a tool would be like and what structures and techniques a tool could express.

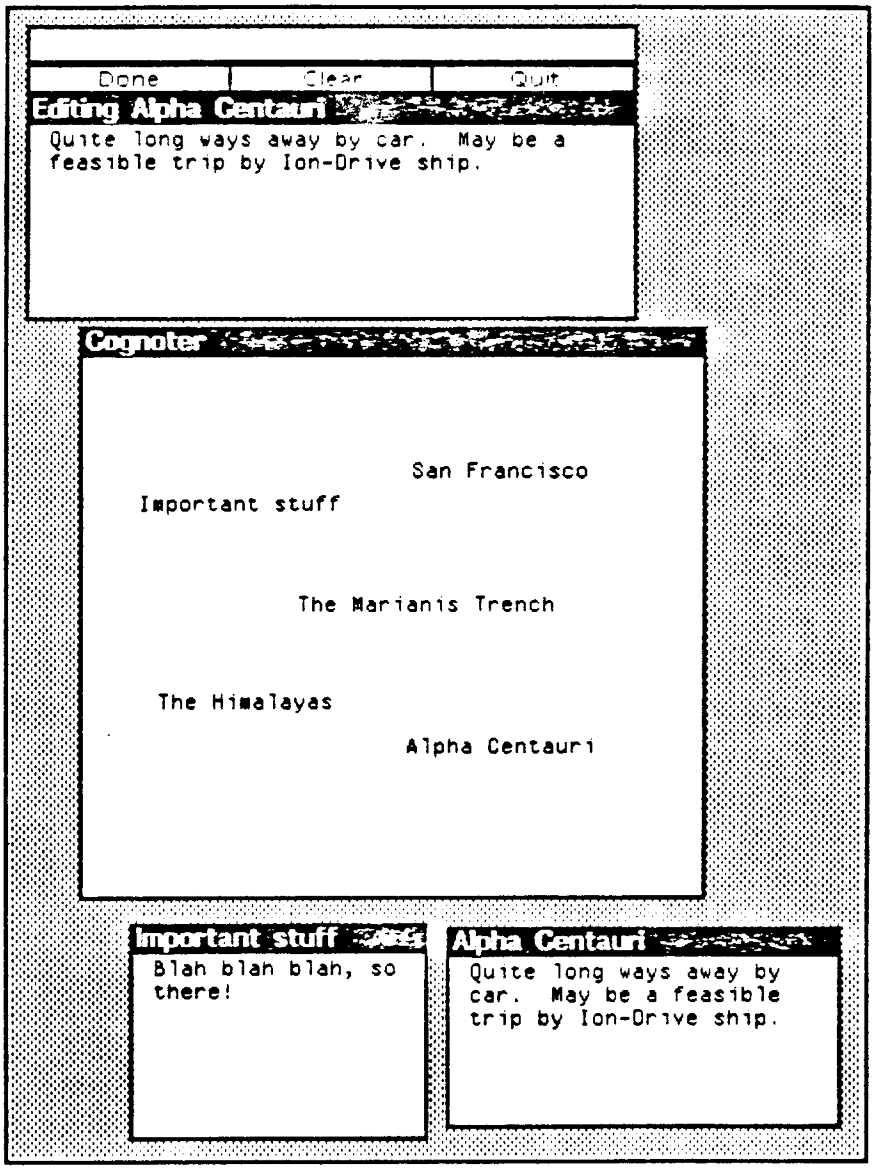
Cognoter is a simple tool to organize ideas for an outline. It's major features are a main window to hold the nodes (short description of ideas), the capability to highlight nodes, move/add/delete nodes, and connect nodes with arrows. A slightly more sophisticated version would allow text to be attached to nodes and have public display windows, and one or more private display/edit windows for adding or displaying comments, further description, and sub-outlines.

Here is the script (structure) for organizing the flow of the cooperative development of an outline:

0. A goal for the outline is chosen.
The group decides to generate an outline for a paper, talk, or design. The main window is created with (let's say) "CoLab paper" as its title.

1. Unconstrained Idea/Node Generation (Brainstorming).
The main window title changes to "CoLab Paper -- Generation". Nodes are freely (asynchronously) placed in the main window by any participants. Node deletion is not allowed in this phase. Criticism and counter-argument is discouraged (though conflicting nodes are possible).

2. Nodes are ordered and clustered.
The main window title changes to "CoLab Paper -- Ordering". Generation may continue but the goal of this phase is clustering the nodes and establishing dependencies. Nodes are moved to be near related nodes. Lines or arrows are added to connected or dependent nodes. A→B might mean "we need A to appreciate B".



```
Quite long ways away by car.  May be a
feasible trip by Ion-Drive ship.
```

```
San Francisco
Important stuff
The Marianis Trench
The Himalayas
Alpha Centauri
```

```
Blah blah blah, so
there!
```

```
Quite long ways away by
car.  May be a feasible
trip by Ion-Drive ship.
```

Cognoter

Figure 1.

3. **Node Connections and Dependencies are considered.**
   The main window title changes to "CoLab Paper – Partitioning" Nodes with no arrows in or out need to be justified or become flagged as candidates for elimination. Nodes that point at nothing may be eliminated or may represent conclusions or open questions. Nodes that are not pointed at may be good starting points (either for the paper or for sections) and should be highlighted.

4. **Irrelevant nodes are excluded and a final order is chosen.**
   The main window title changes to "CoLab Paper – Adjusting". Nodes may now be deleted or regrouped. Some nodes or groups of nodes may be set

aside as interesting but not directly relevant to the current topic. Fine tuning of the ordering takes place: the starting node and section headings are chosen.

5. **The task outline is generated.**
   The outline is generated either by hand or (somehow) automatically. It may be necessary to selectively re-use Cognoter on complex individual outline entries to generate a finer grain in the outline.

Even in this simple tool various dimensions of tool structure need to be considered. For example, in the brainstorming phase do we want input to be put on the screen as it is typed (WYSIWIS)? or do we want to wait until everyone has typed for a while and then merge all the generated ideas. Do we want authorship attached to ideas or should they be anonymous? For more on tool dimensions see section "Tool Dimensions for Meetings and Cooperative Problem Solving".

2.5. Possible Tools

For many applications the three basic phases, "generation – ordering – pruning", is effective. In general, making arguments explicit and guidance (whether by enlightened leader or tool structure) will focus the meeting and avoid circularity[6].

The guiding principles for initial tool designs are utility and immediate interest. We need to supply a tool that people will want to use so they *will* use it. These are some tool designs under consideration.

- *DesignReviewer,* a tool for structuring group design reviews of software. This tool would organize the assumptions, purpose and constraints of a project or program.
- *Argnoter,* a tool for creating and evaluating alternative proposals. It displays multiple proposals and pro and con arguments for each. Explicit goals, assumptions, and evaluation criteria are carried along. If arguments were given validity weights this would also be an opportiunity to explore evaluation functions for automatic ranking of proposals.
- *Chessnoter,* a tool for analysing a chess position. It would need no special chess knowledge other than how to display a position. It would be used to support alternate lines of analysis. This tool would be a good test bed for exploring the capture and representation of alternate lines of reasoning.
- *Negotiator,* a tool for comparing and evaluating 2 (or more) positions in a negotiation. It would display goals, offers, counteroffers, assumptions and proposals.
- *Qualsnoter,* a tool for a PhD student's presentation of a thesis topic. The topic would be entered in a structured manner and explored interactively by the student and a guiding committee.

There are many other possible tools such as a cooperative spreadsheet tool, a tool for administrative assistants, or a Parliamentary Procedure tool.*

3. Previous Work: Computer-based Meetings

There has been little previous work dealing with computer-based meetings. There have been proposals for computer-based group communication systems but only a few working systems have ever been built. Most software "computer conferencing" systems, such as EIES[7], are asynchronous; that is, participants interact at different times.

NLS/AUGMENT was developed at Stanford Research Institute (SRI) by Douglas Engelbart. The main focus of NLS/AUGMENT is collaborative document preparation[8,9,10]. AUGMENT supports many text processing features. "Televiewing" is the AUGMENT feature most relevant to CoLab. It allow screens to be slaved together so everyone in the meeting sees the events on the master screen. NLS and Engelbart will likely be best remembered for the introduction of the mouse pointing device.

Talk in UNIX, *Link* in Tenex, and *OS* (Output Spy) at MIT are system utilities. They synchronously link I/O streams from terminal to terminal or machine to machine. With Talk the connection is for displaying text only. OS allows more complex things like remote debugging.

The meeting system most closely related to our work is RTCAL. *RTCAL/IOLC* (Real Time CALendar / Interactive On Line Conferences) was developed at MIT by Sunil Sarin in 1983[11]. RTCAL allows a group of users to synchronously exchange information from personal calendar databases to schedule a future meeting. The system is synchronous; participants are all on-line at the same time, though usually not in the same room. Both Interactive On-Line Conferencing and CoLab communicate by messages to display objects over a local network. The IOLC prototype that supported RTCAL had a central server. Any changes to a display object were sent to the master server and then images were sent to the clients. CoLab supports non-central control of display objects.†

Single user idea processors and outline tools such as *ThinkTank* are finding acceptance in mass markets. They are single-user tools but are related to our work in that they express structures for problem solving. Their particular strength is allowing interaction with different levels or views of the object (paper, etc.) under consideration or construction.

* A Parliamentary Procedure tool could have the interesting feature that something "out of order" would be seen by no one but the offender. In a courtroom setting the jury would never have to "disregard the comments made by the witness" -- they needn't hear them.

† Sarin notes that a later version of IOLC will support non-central control.

4. Dimensions and Experiments

4.1. Tool Dimensions for Meetings and Cooperative Problem Solving

Structured cooperative problem solving can be more effective and less confusing than the usual committee free-for-alls with dominating personalities, individual fears, fuzzy goals, and circular arguments. A major hypothesis of ours is that structuring group processes will elicit better response from the people involved. There have only been a few attempts at structuring group problem solving and issue exploration. None have been specifically designed with computer support in mind except electronic mail. The *Delphi Method*[12] and *Nominal Group Technique* (NGT)[13] are two structured problem approaching techniques that have been used by governments and corporations. *Electronic Mail* was not intended as a "Group Technique" but it has developed a culture and has been used for serious group problem solving (such as ARPANET use and CommonLisp specifications).

These are some of the structure elements that should be considered in the design of a tool:

- *Simultaneous interaction.* How much simultaneity can we support? It appears to be a good idea to simply let participants interact at will without waiting for "keys" to objects -- if you have a good idea, you want to put it up *now* and let the underlying system keep things straight.
- *Extension of meetings in time and space.* We will need to save some form of a session. Latecomers to a conversation will need to brought up to date. We will want to save the resulting outline or other goal state for later revision or consideration. We may want to save the entire argument structure so other groups can react to it.
- *Monitoring.* There are two kinds of monitoring to consider: internal and external monitoring. *Internal* monitoring keeps records of tool method or option use for later fine tuning and analysis. *External* monitoring could be done with interviews and questionnaires as well as with cameras and taping of sessions.
- *Relaxed WYSIWIS.* WYSIWIS is "What You See Is What I See". We can keep all views of a display object identical, but is this the best thing to do? Computer-based tools can allow various relaxations of WYSIWIS: delayed updating of windows, private windows, remote pointers only on request, associated windows of different sizes or screen positions, different views of the same model, and visible remote cursors only on demand.
- *Anonymity versus Authorship.* Anonymity encourages free interaction; but who gets credit for the useful ideas and how do we follow them up? Authorship gives participants information about the source of an idea, but

personalization of ideas can obscure their true merit.

- *Quality and Quantity of ideas.* Ideally, we want tools that help us generate lots of good ideas and solutions. Generating many ideas seems to imply generation of *some* good ideas. We also need to explore the form and quantity of displayed information for best human comprehension.
- *Conflict Resolution.* How are differences resolved? By system evaluation function? By voting? By leader edict?
- *Highlighting.* Objects and their entries will need to be distinguished at various times. We will consider shading, blinking, wiggling, and musical accompaniment.
- *Structure versus Freedom.*

  "At first sight, the idea of any rules or principles being superimposed on the creative mind seems more likely to hinder than to help, but this is quite untrue in practice. Disciplined thinking focusses inspiration rather than blinkers it." – G. L. Glegg

4.2. Experiments

These are some experiments under consideration for evaluation of CoLab tools and dimensions:

- Build a centerpiece tool with variable parameters and levels of meeting structure.
- Have a competition among alternate forms of a tool (or a tool with user selectable options) to uncover preferences and tradeoffs in utility for meetings.
- Monitor tool use at the Semantic Event level with snoopy software or at the group interaction level by observation.
- Encourage an ongoing forum of tool use and observe which tools and options are most useful.

Other more prosaic (but useful) approaches would be to have heterogeneous groups use and evaluate tools or to have users choose the tools they think they would need for a project and then observe their actual tool usage.

4.3. Evaluation Criteria

The following are the criteria that will be used to evaluate the success of the project and individual tools:

- Tools work in real time.
- Task effectiveness. Ideally the objective effectiveness of the system/tool for a given task would be measureable, but practically this is impossible. User perceived effectiveness and user acceptance will probably have to suffice.
- The relationship of the tools to group problem solving techniques.
- To what extent does a tool's structure guide and enhance meetings.
- Quantity of ideas. More memes[15].
- Quality of ideas. Stronger memes.

In general, formal evaluation of CoLab tools will be difficult. Michael Anderson, a consultant to the Xerox Corporation, has long experience in the evaluation of meeting effectiveness. We hope to bring him in on this phase of the work.

5. CoLab Architecture

There are three basic layers of interest. The lowest layer is the system layer. This includes the actual machines, the Ethernet, input/output devices, InterLisp, and LOOPS. InterLisp is strongly display oriented; it provides software for easy manipulation of windows, menus, bitmaps, and text fonts. LOOPS is a higher level programming environment built on top of InterLisp. It provides object-based, active values, and knowledge-based programming in addition to InterLisp's procedure-based programming. We hope to take the system layer mostly for granted. The next layer up is the software primitive level. Most of our effort so far has been at this level. This includes display objects and CoLab communication protocols. The next layer is the applications layer. Near-future effort will be here. This is the Intellectual Teamwork tools layer. A possible fourth layer is the group problem solving models being tested by the CoLab tools.

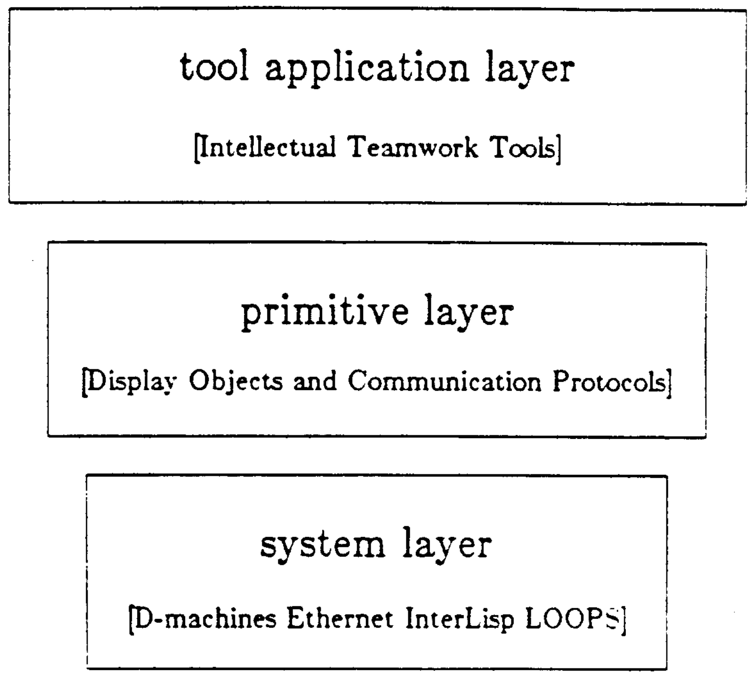
5.1. Software Primitives

CoLab requires a foundation of new software primitives for active and interactive displays. Shared display objects and views of objects must be consistent and synchronized. Network use must be coordinated for densely interacting machines.

Following are some of the existing primitives (for a more complete list, see the CoLab glossary).

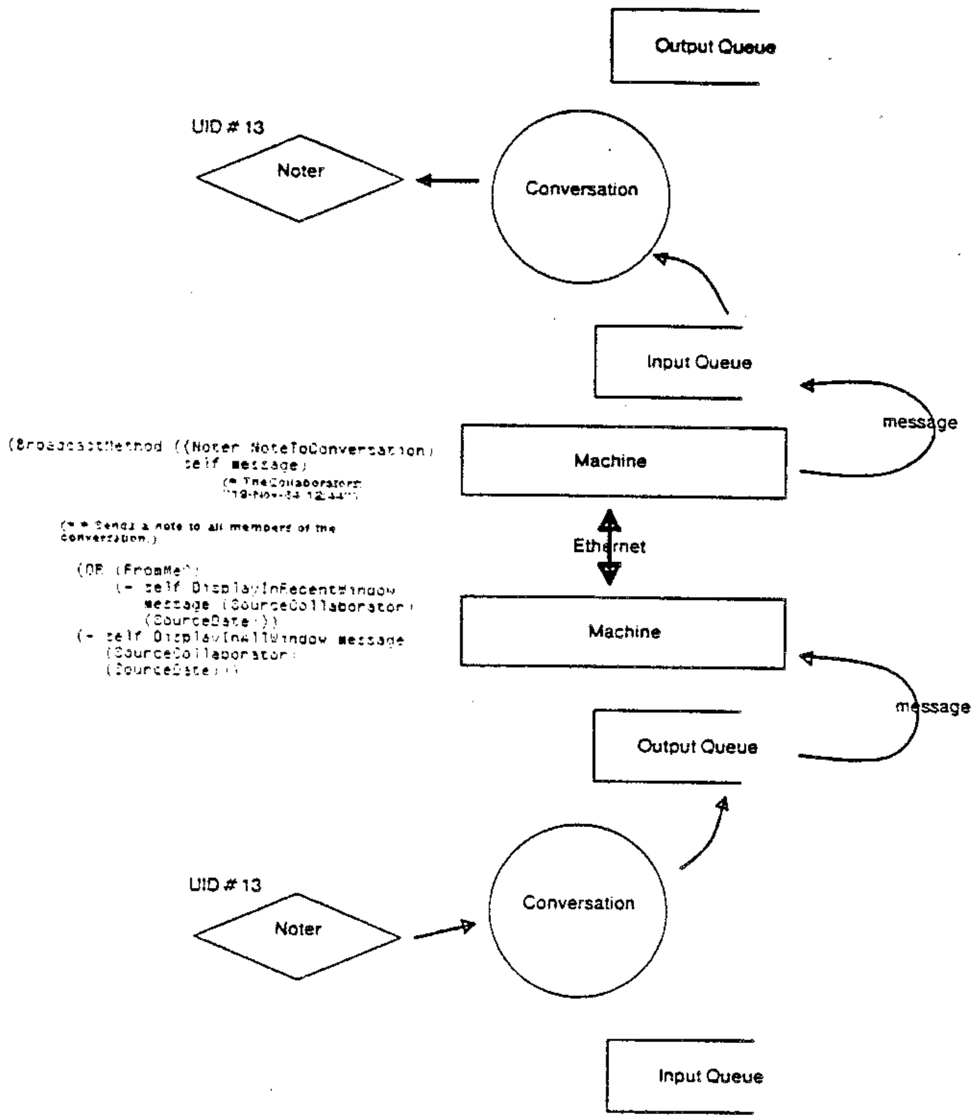
- *RemoteEval* is an InterLisp analog to Remote Procedure Call[16]. It evaluates Lisp expressions in remote environments.

CoLab Architecture Levels


tool application layer
[Intellectual Teamwork Tools]
primitive layer
[Display Objects and Communication Protocols]
system layer
[D-machines Ethernet InterLisp LOOPS]

Architecture Layers
Figure 2.

- *ActiveRegions* are screen display regions that can perceive input events and act on them.
- *Associations* are objects maintained consistently on several machines. A semantic action performed on an associated object will propagate to its associates on other machines.
- *UIDs* are Unique IDentifiers assigned to objects. UIDs are unique across machines, consisting, perhaps, of the machine net address identifier and a timestamp.
- *Semantic Actions* are object changing actions.
- *Conversations* are collections of associations and information about the people and machines involved. Conversations also maintain input and output queues of semantic actions involving its associations or tools.



```
(BroadcastMethod ((Noter NoteToConversation)
                   self message)
                      (* TheCollaborator:
                      "19-Nov-84 12:44")

      (* * Sends a note to all members of the
      conversation.)

   (OR (FromMe?)
         (← self DisplayInRecentWindow
            message (SourceCollaborator)
            (SourceDate)))
   (← self DisplayInAllWindow message
      (SourceCollaborator)
      (SourceDate)))
```

How a Remote Message is Sent

Figure 3.

- *BroadcastMethods* cause semantic actions on a local object to be broadcast to all associates of the object.
- *ColabExec* is the user interface to the high level CoLab environment. It handles some interaction with remote machines and fields local conversation operations, such as adding a new conversation.

- *RemoteMice* are personalized images of mouse cursors active on remote machines.

Figure 3 shows how a remote message gets sent. The lower Noter is given the SendNote message locally. It tells its broadcaster (its Conversation) about the message. The conversation puts the message into its output queue. A process running on the queue finds the message and packs it up to ship across the ethernet. The machines on the other end evaluates the packaged message and this puts the SendNote message into the upper conversation's input queue. The process running on this queue reads the message and finally sends it the upper Noter (with the same UID). The Noter displays the message. The code on the left side is the BroadcastMethod that causes all this to happen.

6. Conclusion

Realistically, the first serious users and evaluators of CoLab tools will be computer professionals. However, computer savvy should not be required. The proposed tools have no command language and will be mouse/menu driven -- "you already know how to use it". The only abilities necessary for useful interaction with the tools are acquaintance with mice/menus, some typing ability (though most typing is done "offline" in a text editor or buffer for presentation when ready), and some knowledge of the text editor used by the system. Time for new users to learn the system will be a tradeoff between ease of interaction and the need to learn to interact in new ways.

6.1. Current Status of CoLab

The software primitives (after a 4th major overhaul) seem to be reasonably stable.* We have reified windows and active regions. Remote message sending and system UIDs are working. Conversations keep their client tools up to date across machines. Keys and locking have been supplanted by broadcast queues, semantic actions, and conflict resolution strategies. Conversations, collaborators and machines can be added or deleted dynamically. The broadcast queues of semantic actions give a decent history of the session -- newcomers can be brought up to date by processing the queues.

Noter, a simple message passing tool, works well. Cognoter and Argnoter are partially developed. Constant RemoteMice are being replaced by Denoter, a request-based pointing facility. We hope to get interesting tools up in the next few months and bring them to a wider audience at Xerox and U. C. Berkeley.

* The software work thus far has been an excellent example of collaborative effort. Daniel Bobrow, Mark Stefik and I have been at it steadily since June 1984. Steve Levy contributed summer 1984 and Ken Kahn and Stan Lanning have been involved this fall (1984).

6.2. Why is CoLab interesting?

In summary, these are the main points of research interest.

- *Extension of meetings in time and space.* We want to save meeting states for future review and for consideration by people not involved in the original meeting.
- *Underlying object communication technology.* The communication display objects, associations, and conversations that allow group use of computers.
- *Structures to organize meetings.* The appropriate structures and levels of structuring for different group problem solving applications.
- *Capture of history.* The capture of meeting history for analysis and the capture of substructures like arguments, proposals, and outlines.

CoLab can be a big step toward enhancing the quality of meetings and therefore at least a small step toward improving the quality of human thought. At Xerox there are anthropologists and psychologists anxious to run experiments and analysis of their own on tools we design. We expect to gain additional insight and new direction from their work.

6.3. Schedule

We need to build an evolutionary system to gain experience so we can build really interesting and usable systems -- this schedule is necessarily tentative.

- *June-August 1984,* the first version of program foundations RemoteMice, Associations, ActiveRegions, Conversations, etc.
- *September-October,* prototype tool design and preliminary testing. Redesign of software primitives.
- *November-December,* testing, evaluation, and redesign of tools and foundations.
- *Winter 1985* the results of this Fall's tests and designs will be used to plan the next phase of thesis work -- the experimental tool(s).

7. Acknowledgements

I am grateful to my research advisor, Richard Fateman, for seeing the potential value of this work and encouraging this departure from previous plans. I am indebted to Mark Stefik, who first conceived of CoLab (though he thought it was called Meeting Lab), for starting me off on such an interesting project. It has been a pleasure to participate in the many discussions, design reviews, and brainstorming sessions with Mark Stefik, Danny Bobrow, and other members of the Xerox Intelligent Systems Lab, especially the Knowledge Systems Area. I would also like to thank the Xerox Corporation for personal support and for providing a stimulating work and programming environment at the Palo Alto Research Center (even though it's too far from Berkeley).

References

1. Robert M. Metcalfe and David R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," *Communications of the ACM,* vol. 19, no. 7, July 1976.
2. Richard F. Lyon, "The Optical Mouse, and an Architectural Methodology for Smart Digital Sensors," Xerox VLSI-81-1, August 1981.
3. Jon L. White, *Evalserver,* Xerox PARC, 1983???. Xerox Lispusers.
4. Larry Tesler, "The Smalltalk Environment," *BYTE,* pp. 90-162, August 1981.
5. D. G. Bobrow and M. J. Stefik, "The Loops Manual (Preliminary Version)," Knowledge-based VLSI Design Group Technical Report KB-VLSI-81-13, August 1982.
6. Mark Stefik. Personal Communication.
7. Starr Roxanne Hiltz and Murray Turoff, *The Network Nation,* Addison-Wesley Advanced Book Program, Reading Massachusetts, 1978.
8. Douglas C. Engelbart and William K. English, "Research Center for Augmenting Human Intellect," Proc. Fall Joint Computing Conference, pp. 395-410, AFIPS press, December 1968.
9. Douglas C. Engelbart, "Collaboration Support Provisions in AUGMENT," OAC'84 Digest, Proc. of the 1984, AFIPS Office Automation Conf., Los Angeles, California, February 1984.
10. Douglas C. Engelbart, "Toward High-Performance Knowledge Workers," OAC'82 Digest, Proc. of the AFIPS Office Automation Conf., pp. 279-290, San Francisco, California, April 1982.
11. Sunil K. Sarin and Irene Greif, "Software for Interactive On-Line Conferences," Proc. ACM-SIGOA Conference on Office Information Systems, June 1984.
12. Harold A. Linstone and Murray Turoff, *The Delphi Method: Techniques and Applications,* Addison-Wesley, Reading, Mass., 1975.
13. Andrew H. Van de Ven and Andre L. Delbecq, "The Effectiveness of Nominal, Delphi and Interacting Group Decision Making Process," *Academy of Management J.*, vol. 17, no. 4, pp. 605-621, 1974.
14. G. L. Glegg, *The Design of Design,* Cambridge University Press, Cambridge, 1969.
15. Richard Dawkins, *The Selfish Gene,* Oxford University Press, New York, 1976.
16. Andrew D. Birrell and Bruce Jay Nelson, "Implementing Remote Procedure Calls," Xerox Palo Alto Technical Report CSL-83-7, December 1983.