Chapter 3

Computer Support for Collaboration

In the preceding chapter, I described an information flow model that identifies several types of information produced by groups. The model provides a lens that lets us penetrate the task domain and see more deeply into the fundamental behavior of groups as they go about the task of building large, complex structures of ideas. As a result, we can characterize groups in terms of generic behaviors that produce those types or transform one type into another. These behaviors are part of a general process of collaboration.

In carrying out this process, groups use a variety of tools. Before the computer, they used paper and pencil, blackboards, and the postal system. After computer and communications systems became widespread, groups adapted these new resources to their collaboration needs — by taking turns editing the same file, by mailing diskettes to one another, and by exchanging files as well as messages through e-mail. Recently, a new generation of tools has begun to appear, designed from the start to support collaboration.

These tools can all be related to the information flow model. Some are used to construct the artifact, others help transform private knowledge into shared knowledge, still others are used to produce ephemeral products. Inevitably, these tools affect the thinking of the individuals and groups that use them. Consequently, the process of collaborative knowledge-construction should be considered in the context of the tools groups use.

Collaborative software tends to fall into one of two large categories: *synchronous* and *asynchronous* tools. Synchronous tools support the simultaneous interactions of two or more members of a group. Shared editors and video conferencing systems are examples.

Asynchronous tools support the independent work of the various members of a group working alone. Electronic mail and distributed file systems are examples of the second category.

The goal for this chapter is to provide a basic understanding of collaboration technology that can be drawn on during the rest of this discussion of *computer-based* collaboration. Because this technology is changing quickly, the discussion emphasizes services provided by categories of systems rather than individual systems, although brief descriptions of selected systems are included to sharpen the image of the class. Thus, it is not intended to be a comprehensive survey of collaboration technology. Near the end of the chapter, one particular system is described in more detail. It combines many of the features found in separate tools and serves as the reference system for the rest of the discussion. The chapter concludes by looking briefly at several issues for research in collaboration systems.

Asynchronous Tools

Asynchronous tools enable collaboration by helping the members of a group work independently on the same tangible product, such as a document, a set of architectural drawings, or the source code for a computer system. Some — such as e-mail, file transfer protocol (ftp), and distributed file systems — are part of the larger system that we routinely work with, and we may not think of them as "collaboration tools." Others were developed from the start with collaborative use in mind.

Virtually all collaboration tools — synchronous as well as asynchronous — depend on some form of underlying computer network to provide communication between workstations or between workstation and specialized processors, such as file servers and supercomputers. Although there are a number of different networks, the Internet has emerged as the de facto standard and the communications infrastructure most collaborative users will have access to. A vast, amorphous federation of separate but interconnected networks, the Internet has become the web that connects the world — or at least those people who use their computers to connect to a network for non proprietary purposes. This includes researchers,

educators, government personnel, people in the military who do not require secure channels, as well as individuals who work for many commercial organizations. It excludes specialized services, such as automatic teller machines, secure military channels, and private networks maintained by individual corporations, such as IBM and Walmart. Consequently, in discussing tools that rely on an underlying computer network, I do so with reference to the Internet. The discussion here covers only a small subset of services; a number of introductions and guides to the Internet describe other services available through it (e.g., Krol, 1992).

E-Mail

By far the most successful collaboration tool is electronic mail. E-mail permits a user connected to a computer network to send a message to another user also connected to the network. One normally sends mail by informing a mail application program of the recipient's name or system ID and his or her network address. The messages one receives accumulate in a file or "mailbox" on his or her local system, which can then be read using a mail application program.

Messages are generally formatted like a memorandum, with header information at the top indicating sender, receiver, date, subject, and so forth, followed by the text of the message. The message can be typed directly, or the sender can also copy a computer file into the body of the message. Thus, collaborators can exchange comments on a document they are working on, or, using the file copy option, they can exchange entire versions. Of course, files for other types of information in addition to text can be sent.

Although different networks use different address formats, most now recognize the form supported by the Internet. Because many of these same networks are physically linked to one another, through so-called *gateways*, it is now possible to send e-mail to someone located virtually anywhere in the world, have it relayed from network to network, and end up in that individual's local mailbox. Thus, it is almost as easy to send a message to a collaborator on another continent as it is to someone across the hall.

Although e-mail's potential as a tool for remote collaboration is obvious, the ways it has been used for that purpose have not always been what was expected. When e-mail was first introduced, many

people thought that one of the strongest effects would be to facilitate collaborative relationships among people located at a distance. Indeed, this effect has been observed but, in some cases, not as strongly as expected. In a recent study done by the RAND Corporation, researchers found that people rarely sent messages to people at distant locations, and nearly half of the messages sent were to people in the sender's immediate vicinity (Bikson & Eveland, 1990). The more important effect, they speculate, may be overcoming temporal, rather than spatial, barriers. Whereas individuals may not be available for a visit or telephone call, they can accomplish a number of interactions over a day through sequences of e-mail messages and responses.

E-mail can also affect the ways groups organize themselves. In organizations where individuals participate in multiple work groups, e-mail tends to increase the number of teams an individual participates in (Bikson & Eveland, 1990). Similarly, it tends to spread leadership roles more evenly across the pool of participants. It also results in fewer face-to-face meetings, telephone calls, and other traditional forms of communication, although face-to-face meetings were preferred for resolving conflicts, reaching consensus, and addressing other complex and/or sensitive issues. There is also evidence that groups that make effective use of e-mail tend to be more successful than groups that do not (Fineholt, Sproull, & Kiesler, 1990).

FTP

Although e-mail can meet many of the communications requirements of collaborators, there are some needs for which it is not well-suited. For example, it is easy for someone to send a file he or she has been working on to a collaborator, but it is less straightforward for someone who wants a file located on another person's system to get that file. One can always send an e-mail message to the owner and have that person mail the file back. But that requires considerable human intervention. Many times, one would rather copy a remote file directly, without having to ask someone to send it.

This can be done using an Internet facility called ftp, an abbreviation for file transfer protocol. Whereas e-mail is sender driven, ftp is receiver driven. Ftp allows the person who wants a file stored on a remote machine to have that system send a copy of the file

to his or her local machine, without involving another human being. This is done by starting an application program, called ftp, on the local machine and then contacting the machine where the desired file is located, through the Internet. Once a connection with the remote machine is established, a similar ftp program is started on that machine, as well. One can then send files back and forth between the two machines, subject to several constraints. The user must have the permission of the owner of the file to copy it, know the name and location of the desired file, and be a registered user on both systems.

To get around these restrictions, many sites have established anonymous ftp directories in which various public documents — such as technical reports, program documentation, and even computer programs — are placed. Anyone may access these files, and they are stored in the same logical location on all systems. A number of specialized collections of information are beginning to appear as anonymous ftp directories, ranging from aeronautics to zymurgy (Krol, 1992).

Thus, private ftp is well-suited for small groups of collaborators who wish to share a specific set of files, whereas anonymous ftp lends itself to less structured forms of cooperative work where the user population is larger and not necessarily known to one another.

Network File Systems

Although ftp and other forms of explicit file transfer are extremely useful, they still require considerable human overhead. Users must know the name and location of the system where the desired file is stored, invoke a file transfer program on both the local machine and the remote machines, and assume responsibility for maintaining consistency between the original and the copy, if that is a concern. These problems are solved and a number of additional services provided by network file systems (NFS). Examples include Sun's NFS (Walsh, 1985), DECnet (DEC, 1985), and the Andrew file system (AFS) (Spector & Kazar, 1991). Because AFS appears to be emerging as the de facto standard, at least for the UNIX and Internet worlds, and because it provides services not yet available in other systems, it serves as the example system for this class.

To understand how AFS and other similar systems can help groups work more easily with a common body of material, one must

first understand the naming conventions used by various kinds of file system. Individual computers provide users with a hierarchy of directories (*folders*, in Apple terminology) and files. One can think of this hierarchy as a tree or organization chart, with the so-called root directory at the top. The root directory contains a set of other directories and/or data files. Each of these directories, in turn, can contain other directories and files, and so on. Because the directories and files have names, as well as the particular computer, one can identify any given item stored on that system by listing the names of all of the directories one must go through, from the top down, before reaching the desired item. For example, on my personal computer, this chapter of the book might be referred to as: */jbs/docs/CI/chapter3*, where *jbs* is the name of my computer, *docs* a directory of documents, *CI* the directory for this book, and *chapter3* the current chapter. The overall hierarchical structure of names is sometimes called a *name space.*

Network file systems extend the name space to a collection of machines. The first extrapolation is to machines located on the same local area network (LAN) or collection of interconnected LANs. Such configurations can be found in academic departments, corporate sites, or other similar geographic and administrative centers; they can contain hundreds of computers, including both individual workstations and specialized file servers. Network file systems such as AFS permit such collections to share a single logical hierarchy of files that integrates the files stored on all of the individual machines in the network. This name space is similar to that described previously, but has an additional tier added to the top. The root of the tree now refers to the site or collection, as opposed to the individual computer, whereas the second level names each machine in the collection. In this expanded name space, the example file might be referred to as */unc/jbs/docs/CI/chapter. unc* is the site, whereas the rest of the name remains the same.

This capability is obviously useful for collaborative groups. For example, a project can set up a system of directories and files that comprise the artifact, as discussed in chapter 2. Each member of the group could then view and edit this material, add and delete files, and so forth. Instead of having to consciously move files from one machine to another, as required by ftp, users can access files stored on any of the machines simply by referring to them by name. After that, the system does the rest. Of course, the system leaves it up to the group to keep up with logical relationships among files and directories

(e.g., which files constitute the chapters of a book or the current version of a system), to maintain internal consistency within the data, to ensure that one member's changes do not undo the work of another, and so forth. Nevertheless, network file systems make it much easier for a group of collaborators to work with a shared collection of material.

A wide-area file system extends the name space one step further to include any machine or site that is accessible through the Internet and runs the wide-area file system in the local environment. This is done by adding still another tier to the top of the name space tree. In this expanded name space, the example file might be referred to as */afs/andrew.unc.edu/jbs/docs/CI/chapter3*. *afs* refers the entire structure — in this case, all of the various Andrew file systems located at different sites that have been drawn into a single logical structure — *andrew.unc.edu* is the Internet name for one particular site — unc — whereas the rest of the name below the site remains the same. Andrew then provides access to any file in this *entire* (potentially) worldwide system in exactly the same way it provides access to a file in the same local area. Thus, collaborators located anyplace in the world can work on a common set of files with virtually the same convenience they could were they located in the same building, so long as they have access to the Internet. The only differences are the longer file names and the additional time needed for access, caused by network delays and system overhead.

Collaborative Applications

The systems described previously are parts of the basic computing infrastructure. They provide general services, such as exchanging messages and transferring or accessing files. Applications, on the other hand, provide functions that relate more directly to substantive aspects of the task at hand, such as a text editor being used to write a document or a drawing program for drawing a diagram. Most applications designed explicitly to support collaboration are synchronous, such as shared editors or meeting support systems. Three major classes of these tools are discussed in the section that follows. However, a few asynchronous applications have been built that support asynchronous work, primarily collaborative writing and/or document management. In this section, I briefly describe

several of these applications to complete the discussion of asynchronous tools.

One particularly innovative application is a writing tool developed at Carnegie Mellon University, called the "work in PREParation" editor or, more commonly, PREP (Neuwirth, Kaufer, Chandhok, & Morris, 1990). It uses a two-dimensional grid, similar to a spread sheet, to enable the members of a group to work individually on a document they are co-authoring. The core text is assigned a column and is divided into horizontal segments, typically one paragraph per cell. Each participant is then assigned a different column in which he or she can comment on, rewrite, or otherwise edit the core text. Members can also comment on each other's comments, as well. Thus, one can scan horizontally across the display and see all of the comments and revisions for a given section of the text or scan vertically to see all comments and revisions made by a given individual. As might be expected, PREP has proved to be most useful for dealing with small-grain writing problems — such as style, sentence clarity, and paragraph structure — and less so for large-scale problems — such as overall organization and document structure.

Although PREP provides a structured environment with respect to the document and group members' comments and revisions, it does not impose structure with regard to members' social roles or authority. This design philosophy contrasts with that of Quilt, a collaborative writing system developed at Bell Labs (Leland, Fish, & Kraut, 1988). Quilt takes the opposite approach by encouraging groups to establish well-defined social and authority structures with respect to co-authored documents. The system includes three participant roles (co-author, commentator, and reader), six types of documents or comments (base document, suggested revision, public comment, private comment, message, and history), and a set of functions or actions users may employ (create, modify, delete, attach a suggested revision, attach a public comment, attach a private comment, attach a directed message, and read). Like PREP, Quilt enables users to attach comments to documents but through a form of hypertext link that must be followed by a reader to see the comment. One nice function found in Quilt that is not part of PREP is the capability to attach messages that invoke e-mail responses, such as a reminder to someone to look at a particular section of a document.

An innovative asynchronous application that is not a writing tool per se is Oval, an acronym for *objects, views, agents,* and *links*, developed by Tom Malone and his colleagues at MIT (Malone, Lai, &

Fry, 1992). Oval is a meta-application that enables users to create their own specialized cooperative work applications. They do this using Oval's four basic components. Objects are collections of attributes that can be defined by the user, such as a form that includes names and addresses. Views present particular representations of objects or collections of objects. Agents are sets of rules that can operate on objects and the other components without the direct attention of the user. They can perform a variety of actions, such as deleting objects older than a certain date or sending e-mail to the user to alert him or her of a change made to an object by someone else. Links provide a mechanism to build structures of objects. Such structures can be used for a variety of purposes, from composing documents from collections of objects to providing the paths along which agents move to carry out their actions. As a test of Oval's generality, the developers built emulations of several well-known CSCW applications — gIBIS, Coordinator, and Lotus Notes.

Lotus Notes combines features of both asynchronous collaborative applications, such as the two writing systems just described, and the network file systems, described earlier (Lotus, 1993). Notes currently supports three basic functions: e-mail, text editing, and document management. The e-mail and word processing functions are similar to those in other systems; it is the document management component that sets Notes apart from other applications. Built to run on a network of PCs, Notes enables a group or organization to share a collection of documents that all members may access through the network and work on concurrently, within the constraints of security and a system of permissions. For example, a member of a group can access a document, work on it, and return it to the storage system. She can then send e-mail to a colleague telling him that it is his turn to work on it. The second person can, in turn, access the newly revised document and do likewise.

Currently, Notes works best within local area networks, where all members may access the latest version of a document. For groups distributed over a wider area, data must be copied from their primary storage locations and replicated on remote systems. Users at remote locations, thus, work with replicated versions of documents, not the "originals." Because changes to replicated data are updated among the various storage systems only occasionally, such as overnight, close coordination of work within widely distributed groups is difficult. Thus, Notes provides some of the function found in network file

systems but with limited update and support for a limited number of data types.

Taken as a group, asynchronous collaboration tools and systems tend to be used with a group's more tangible forms of information. Because ftp transfers data from one file system to another, to the extent the files are a group resource, they represent some part of the artifact — either instrumental or target products. Private files obtained through ftp would represent either information outside the scope of the project, or, if the data contribute to a member's work on the project, they would become instrumental products. Network file systems have a similar relationship to the information flow model because they provide essentially the same access as ftp, but much more conveniently and with fewer restrictions (e.g., not requiring a login on remote systems). The same is true for collaborative applications, because they are used for direct work on the artifact.

E-mail, however, is used with all three major types, although not necessarily in equal proportion. Because most groups do not systematically save e-mail correspondence, messages tend to be ephemeral. But e-mail is also used to send copies of files between members and, thus, serves as a communication vehicle for both instrumental and target products. It can also affect intangible knowledge. To the extent that a message is substantive, it contributes to an individual's private store of intangible knowledge or the knowledge the receiver shares with (at least) the sender of the message. It can also contribute to knowledge shared by the group as a whole. For example, it is not uncommon in software development projects for individuals to make general announcements by e-mail, such as informing the group of a particular problem or that a new version of a module is available.

Thus, although asynchronous tools tend to be associated with tangible and ephemeral forms, some play important roles with regard to intangible knowledge as well.

Synchronous Tools

Synchronous tools enable collaboration by permitting two or more people who are separated geographically to interact with one

another through a communication network. A large and diverse set of tools fall into this category. Some are concerned primarily with supporting verbal and visual interaction, such as two people engaged in a private conversation or a group having a meeting. Others enable several people to work together at the same time on the same tangible product, such a co-editing a document or drawing. In this section, three types of systems are discussed: audio/video conferencing, shared applications, and meeting support tools.

Most synchronous tools create working situations for distributed groups that approximate one or more "natural" activities that occur in groups whose members are all located at the same site. A few try to create new forms of interaction, based on new metaphors, such as two colleagues sitting opposite one another drawing on a vertical pane of glass that stands between them (Ishii, Kobayashi, & Grudin, 1992). However, they are the exception. Consequently, most tools are discussed in relation to the proximate forms of interaction they simulate.

Audio/Video Conferencing

Audio/video conferencing tools allow members of a group to carry on informal conversations as well as more structured discussions at a distance. There are numerous analogs for this type of interaction: two people talking in an office, several colleagues having a discussion over lunch, a group holding a meeting in a conference room, or a programmer asking a question of a colleague sitting at a nearby workstation. Most such interactions have to do with intangible knowledge or with the social and administrative aspects of a project. Sometimes, they may also involve ephemeral products, such as a presentation that includes visual material. Occasionally, the group may combine verbal and visual interaction with work on the artifact, such as a group conducting a walkthrough of a new segment of code. But, by and large, these activities do not constitute direct and immediate work on the artifact; rather, they contribute indirectly — but essentially — to its development through the building of shared knowledge, through coordination, and by helping groups build and maintain a web of personal relationships and impressions.

The oldest, and by far the most widely used, tool to support verbal interaction at a distance is the telephone. It is so familiar and

so taken for granted that most of us do not think of it as a tool for collaboration. But, we should not underestimate its importance for that purpose or as a standard against which to measure new alternatives.

To state the obvious, the telephone is most effective as a tool for two-way conversations. Many people use it reflexively for that purpose. Although they may be vaguely aware that talking on the phone and talking face-to-face involve different social and intellectual dynamics, most can do either with little or no conscious attention to the medium of interaction. Consequently, when two collaborators at a distance need to talk to one another, the telephone provides a level of service that is hard to beat.

We begin to notice the phone as medium when we use it in nonroutine ways. For example, calls that travel over great distances, such as between the United States and India or Australia, are often routed through a satellite. This can introduce a delay of nearly a second between the time one person says something and the other hears it. This latency significantly alters the natural rhythms of conversation. For example, the two speakers often talk over one another; when they realize this, each stops and waits for the other; hearing nothing, both respond, again talking over one another. Most people eventually adapt to the medium, often by exchanging rather formal statements and by signaling when each statement is complete. But, for many, it is not a natural or comfortable form of conversation.

A second nonroutine use of the phone system is for group discussions. Conference calls allow three or more participants to take part in a conversation. Each hears what the others say, and each may speak up at any time. But it is not the same as being in the same room. Because the group cannot see one another, they do not have access to visual cues, such as gestures or facial expressions, that signal when someone is about to say something, color what is said, or identify the speaker. Thus, although conference calls represent an important resource for distributed groups, the form of interaction this technology enables is different from same room interaction and leaves much to be desired.

To get around these problems, but also to provide additional capabilities, some groups have supplemented audio communication with real-time video channels. One of the earliest demonstrations of this was given by Doug Engelbart at the 1968 Spring Joint Computer Conference in San Francisco (Engelbart & English, 1968). During a

plenary session, he and a colleague located some 30 miles away carried on a joint work session that included accessing a common store of information, editing a document together, and talking to and seeing one another through live, two-way audio/video channels. Much recent work in teleconferencing systems has been an attempt to make this mode of work more widely available.

In 1985, as an experiment in distributed organization, a group at Xerox's Palo Alto Research Center (PARC) was divided into two subgroups: one remained at PARC and the other was moved to Portland, Oregon (Abel, 1990). In addition to e-mail and high-speed data communications, members at the two sites were able to communicate with one another through "video walls" located in commons areas, such as lounges and coffee rooms. These set-ups consisted of large displays (although not really "wall" sized) and accompanying cameras and microphones, creating a form of "shared cross-site space" in which members could congregate and talk informally with one another. A similar system, called the VideoWindow, was developed about the same time by Bellcore (1989; Fish, Kraut, & Chalfonte, 1990).

Audio/video setups such as these seem to be most useful for informal discussions and for helping members of a group develop personal impressions of one another; however, they are less well suited for private, highly technical, or lengthy discussions. Consequently, during the latter stages of the PARC experiment, monitors and cameras were also installed in individual offices (Abel, 1990). Bellcore's CRUISER system was similar but used the computer to switch the video from one location to another, enabling users to browse one another's offices — located on several floors of a large building — as if they were walking, or "cruising," down a hallway (Fish, 1989; Fish, Kraut, Root, & Rice, 1992; Root, 1988). A more ambitious project was recently begun at Xerox's EuroPARC. They developed an extensive video environment, which they call *media space,* that includes each office at the Ravenscroft site as well as selected offices at Palo Alto (Gaver et al., 1992). What sets this system apart is the extensive set of applications they developed to supplement basic audio/video communication. They included a central calendar system that sets up video meetings and lists current or future meetings, CRUISER-like "hallway" browsing, and pages of "postage stamp" images of colleagues produced by slow-scan video updated every 5 minutes or so. This last application lets the group see at a

glance who is around — within their 6,000 mile wide media space — and what they are doing.

To study the effects this technology on users, researchers at the University of Toronto have developed a system, called CAVECAT (Computer Audio Video Enhanced Collaboration At Toronto), which they use in both their own work and in more formal studies (Mantei et al., 1991; Sellen, 1992). The typical office set up of cameras mounted over participants' workstations with monitors located to the side does not provide realistic eye contact or a natural sense of spatial orientation. To counteract these effects, the Toronto group has developed an elegant device they call *hydra*. Each hydra unit, which resembles a carton of cigarettes mounted vertically on a swivel base, contains a speaker, a microphone, a color monitor, and a video camera located a fraction of an inch from the monitor. Because the camera and display are so close to one another, when a person looks at the display, he or she seems to be looking directly at the camera, creating strong eye contact with the viewer. During a meeting, several units can be placed on the desk in front of each participant to provide a sense of spatial orientation.

All of these are research or experimental systems, implemented using cable video technology and leased, high bandwidth communications lines. Consequently, they are expensive, one of a kind systems. Although still not cheap, equipment aimed at the corporate and consumer markets is beginning to appear. One such device is the codec. A rather compact piece of equipment, it uses digital compression to send video and audio signals over conventional dial-up telephone lines, although not yet at the NTSC standard 30 frames per second, making it possible to hold teleconference meetings in conventional conference rooms. AT&T has recently introduced three new forms of its Picturephone. Priced at approximately $1,000, these devices can be placed on individual desks or even in the home. Although this technology could make two-way video/audio communications much more widespread, we should be cautious in our expectations, given the unenthusiastic reception of the original Picturephone, introduced at the 1964 World's Fair.

To put this technology into perspective, I suspect that in the long run teleconferencing will prove useful in proportion to the computer applications that are available to control it, schedule it, and to provide services that go beyond simple "talking heads." If so, this would mean that the network that supports video/audio communication must be integrated with the group's computer network. This could be done in

two ways. First, the video/audio signal could be compressed and transmitted as digital data over a conventional computer network, such as the Internet. Several computer applications are available that support telephone-like conversations over the Internet, but with considerable distortion.

Currently, large blocks of data are divided into pieces and sent over the networks in the form of packets. This is like transporting a large group of people from a hotel to a reception by putting small groups of them into individual cabs — they may all eventually get there, but not necessarily at regular intervals or in the order in which they left the hotel. Delays and irregularities in the arrival of video/audio information create distortions and latency. Given the intrusive effects of latency in distant phone conversations, as discussed earlier, this could severely limit or compromise the kind of audio/video communication that can be provided through current computer networks. One answer to this problem lies in advanced *isochronous* network architectures that provide pulse-like communication in which packets of information are delivered with guaranteed regularity. Although this technology is promising for widespread, high quality teleconferencing, using it for that purpose will require an extensive supporting infrastructure that rests atop the network's basic transport facilities. In effect, this would amount to developing a second telephone-like network within an upgraded version of the Internet or its successor.

An alternative approach is to develop more extensive connections between computer networks and the telephone network. Telephonic networks are circuit based, providing a virtual wire between endpoints with guaranteed bandwidth and latency characteristics. Thus, one can purchase one grade of circuit for two-way telephone conversations and another grade of circuit, with greater capacity, for full-motion video. At present, higher grade services are limited, costly, and awkward to set up and take down. But as fiber optic technology becomes widespread, network capacity will increase dramatically, and advanced telephone switches could provide dial-up high-capacity circuits adequate for teleconferencing, similar to current dial-up audio circuits.

If such an infrastructure were in place — and it seems probable that it will be relatively soon — it would be possible to build seamless collaboration environments that implement some of their services using conventional data networks, whereas other services, such as teleconferencing, would use telephonic networks. Each technology

would be responsible for providing the services it does best. Although the current gap between data and telephonic networks seems large, most wide-area data networks are implemented using leased telephone lines — they simply attach specialized hardware that supports a different network protocol to high bandwidth telephone lines. Thus, dual-purpose networks could be built using the same underlying physical infrastructure.

As we look to the future, we should expect teleconferencing to become an increasingly available resource for collaborative groups. At first, it will be provided in separate systems; but, once established and once the connection is made between it and the computing infrastructure, a number of new applications should appear that will make video/audio communication useful in ways we cannot yet imagine. To make this happen, however, will require as much work in matters legal and political as in technical development.

Shared Applications

Shared applications permit two or more users whose workstations are connected by a communications link — such as a LAN, the Internet, or a telephone/modem connection — to work simultaneously and interactively with one another on the same data. Multiple users can provide input to an application program and see the results of that input on their respective displays. This mode of work is roughly analogous to several individuals sitting down at the same workstation, where they can all see the same display and where they can take turns using the keyboard and mouse. Normally, only one member of the group at a time is the active user — his or her workstation is the one that currently provides input to the system — while all of the others are passive users — they see the results of the active user's input on their respective displays. Most systems include floor control mechanisms that allow individuals to request to be the active user and for control to be passed among the participants. A few permit multiple active users in a free-for-all mode of work.

A typical task for which a group might use a shared application is co-authoring a document. The shared application could be a word processor or text editor. The group would set up one or more computer conferences in which the members participate from their individual workstations. During these sessions, all participants would

be able to read the same segment of the co-authored document on their respective screens, and they could take turns writing or editing sections.

If during the session one member wants to make a comment to the others, he or she could type the comment into the document for the others to read. Some systems include chat windows, in addition to the shared application, that keep this type of informal interaction separate. More desirable would be to include audio and video conferencing, described in the preceding section, as an integral part of the shared application; but, currently, the two activities tend to be supported by separate and distinct systems. Thus, shared applications facilitate work on tangible products or, if the group does not save its work, ephemeral products, but they are awkward for group interactions concerned with intangible knowledge. As a result, the analogy of several people sitting down at the same workstation holds fairly well with regard to interaction with the application but breaks down with regard to the conversation such a group would normally carry on while they worked.

Although computer conferencing systems share many features, they differ from one another in subtle, but important, ways. To see these distinctions, let's look briefly at several example systems.

Aspects is a commercially available shared editing system that runs on Apple Macintosh computers using an Appletalk network or telephone lines with modems (Group Technologies, Inc., 1990). It includes three separate applications — word processing, drawing, and painting — that can be used separately or together in joint editing sessions. The three strongly resemble early versions of MacWrite, MacDraw, and MacPaint. Consequently, the price that must be paid for Aspects' collaboration support is accepting its very basic applications, as opposed to more sophisticated applications — such as MSWord, MacDraw Pro, and SuperPaint — members may be accustomed to using for individual work.

Aspects is an example of a *closed architecture system.* Other closed collaboration systems include ShrEdit, a collaborative text editor used primarily for studies of group behavior (McGuffin & Olson, 1992), and SEPIA, a system that includes planning, argumentation, as well as editing modes (Streitz et al., 1991). These systems provide fixed sets of tools, and the user is limited to those and only those tools. Should users want to work together using a different application, such as a project management tool, a scheduling program,

or even a different text editor or drawing package, they could do so with these systems.

In contrast, *open architecture* conferencing systems provide general mechanisms that allow groups to work collaboratively with thousands of existing single-user applications, rather than with only the handful of specially tailored programs provided by closed systems. These systems currently operate within the UNIX and X Window environments and, as a result, are often referred to as *shared X* systems. The reasons for this are technical: the X Window architecture makes it convenient to intercept in a general way the input from mouse and keyboard directed toward an application as well as the output going from the application to the display. The conferencing system can then selectively redirect any given conference participant's input to the application, while blocking input from all the others; conversely, it can broadcast the output from the application to all participants' workstations. Thus, virtually any existing X/UNIX application can be used collaboratively, as well as new applications that will be developed in the future. Examples of shared X systems include Rendezvous (Patterson, Hill, & Rohall, 1990), MMConf (Crowley, Milazzo, Baker, Forsdick, & Tomlinson, 1990), Rapport (Ensor, Ahuja, Horn, & Lucco, 1988), and xtv (Abdel-Wahab, Guan, & Nievergelt, 1988).

Although shared X systems provide generality in the sense that a large number of single-user applications can be used collaboratively, they also impose restrictions. For example, all users see exactly the same display during conferenced sessions. Should one user want to scroll the window up to data not currently visible, he or she could not do so without first becoming the active user and then changing the information shown on all participants' workstations. An alternative would be a conferencing system that lets a group share a given application and data but lets each user control his or her display separately. Thus, a group could take turns editing a document, but while one person is editing, the others could look at different parts of it. Another possibility would be to allow multiple editors so long as two people do not try to edit the same paragraph. To support more flexible interactions such as these, Prasun Dewan developed a set of tools he called Suite for building advanced collaboration-aware applications (Dewan & Choudhary, 1991). Suite treats major collaboration design issues — such as the frequency users' screens are updated, the granularity of the data that are shared, and the restrictiveness of the coupling — as parameters that can easily be

changed, permitting system designers to test different models of conferencing. Future systems could allow users themselves to control these parameters and thereby customize the system to suit their preferences.

To put conferencing systems into perspective, they represent a break with traditional systems in their goal of supporting groups working in close, continuous interaction on the artifact, as opposed to individuals working alone on separate data. For some activities — for example, a walkthrough of computer code, a review of a document, or a markup of legislation — working in a distributed manner using a conferencing system may actually be preferable to working conventionally in the same room. But this has not been demonstrated. Nor is it known with certainty for which activities computer conferencing is most appropriate, how those activities fit into the overall collaborative process, or the incremental benefits to be gained by adding human communication channels — audio alone and audio coupled with video. Thus, it is a technology that offers great promise but is still in its infancy.

Meeting Support

Although computer conferencing systems are quite flexible and, thus, can be used in a variety of settings, they are oriented toward activities in which participants work collaboratively from their respective offices while doing some type of group editing task. A different set of tools has been developed to support situations in which participants gather in the same room at the same time to take part in a single, interrelated activity, such as a meeting.

A key part of most meeting support systems is a large, electronic display with facilities for various members to write, draw, or show computer and/or video data on the display. The early trend was toward setting up dedicated, specially constructed rooms. The display was provided by rear projection of video or computer data on screens mounted permanently in the walls. Participants controlled the display from individual workstations, often built into specially constructed conference tables or large, forum-style desks. Recently, a more flexible form of display has emerged that can be used in conventional meeting rooms or public spaces without physical alteration of the room.

Early meeting support systems tended to divide into two types. One group was oriented toward small meetings, typically involving six or eight participants, that would normally be held in a conference room. Most of these systems were built during the mid-1980s. Three examples are Capture Lab (Mantei, 1988), Xerox PARC's electronic meeting room built as part of their Colab project (Stefik et al., 1987), and a similar facility at MCC (Cook, Ellis, Graf, Rein, & Smith, 1987). The software provided by these systems ranged from simple editors, used off-line to enter text that was then cut and pasted to a display window, to relatively sophisticated hypertext systems that provided a spatial field on which the group could collectively build two-dimensional structures of ideas.

The expected mode of work was for individual members to take turns recording ideas on one of the workstations; the result would then be shown on the large display. Because participants were seated around a special conference table with embedded workstations, they could also see one another and talk both about ideas to be recorded on the group display as well as what was already there. Although developers attempted to minimize the effects of both hardware and software on behavior, all report significant changes in group dynamics. For example, if while one person was talking another began typing information on the screen, the group's attention often shifted to the display, including the speaker's who would sometimes forget what he or she was saying (Mantei, 1988). Thus, the display tended not only to focus discussion but to dominate it.

A second type of system is oriented toward large meetings involving several dozen participants. A well-known example is the group decision support system built at the University of Arizona (Nunamaker, Dennis, Valacich, Vogel, & George, 1991). Located in the School of Business, it supports tasks such as long-term strategic planning and policy formulation as opposed to small group discussions. The Arizona room, itself, has the feel of a corporate setting in its decor, lighting, and double rows of raised, forum-style desks that accommodate the 24 workstations used by participants. In addition to large display projection, the system also includes a number of applications that support group brainstorming, organizing, voting, and policy formation.

The typical mode of work is for participants to type messages into a shared database that can be read by all, send messages to authors of particular comments, and otherwise engage in multiple simultaneous on-line conversations. Thus, there is less emphasis on conventional

discussion and more reliance on the system for exchanging views. However, at critical points, on-line activity can be interrupted, and the group can discuss a particular comment or data shown on one of the large displays. After that, the group may continue the current task, using the same program they were working with before the interruption, or they may move to a different task, using a different program. Implicit in the system, then, is a social/cognitive model of group intellectual behavior that encourages the group to move from broad, open-ended consideration of an issue, toward closure, to the drafting of some tangible product that records their collective thinking.

Recently, a third type of meeting support technology has appeared that is less intrusive and can be used in conventional conference rooms. Whereas the systems described previously assume that input is provided by a keyboard and, perhaps, a mouse, these systems are pen-based and assume that handwriting and hand-drawn figures are the major forms of input. Three examples include Liveboard, a custom hardware–software configuration developed at Xerox PARC (Elrod et al., 1992), Smart Technologies' System 2000 (SMART, 1993), and a research system developed at the University of Flinders (Mudge & Bergmann, 1993). However, just as *xerox* has become the generic name for copiers, *liveboard* is becoming the generic name for stylus-oriented systems based on the whiteboard metaphor. Consequently, I focus here on Xerox PARC's version of the concept.

Although Liveboard's developers look forward to flat, wall-mounted displays the size of actual whiteboards, currently, the unit is a large piece of furniture roughly the size of a console TV but twice as high. It provides an approximately 3 feet by 4 feet surface on which computer output can be displayed by a self-contained rear projection unit and on which users may "draw" using a special cordless optical stylus. The computer in the system is a conventional workstation — a UNIX workstation in research units, replaced by a PC in the commercial version. Although bulky, a Liveboard can still fit into the corner of many existing conference rooms and public places, such as lounges and coffee rooms, without physical alterations to the building.

The expected mode of work is similar to that for conventional groups meeting in conference rooms and using a conventional whiteboard. For example, the group might sit around a conference table discussing a problem or issue, while one or more participants stand in front of the display and write or draw on it with the optical pen, just as they would stand in front of a whiteboard and draw on it

with a felt marker. Currently, some half-dozen special purpose applications have been written for the Liveboard, including meeting tools, several whiteboard applications support for slideshow-like presentations, games, and a general interface to its internal workstation. However, because the Liveboard screen can be used as a conventional workstation display, the group can also access documents or other products stored in computer files and work with them using conventional applications. In the future, it should be possible to connect several Liveboards to one another through a computer network to support distributed meetings, although such meetings will, no doubt, require audio and, perhaps, video channels, as well.

The liveboard concept is a very interesting development. Designers seem to be on the right track to stress flexibility and hand, versus keyboard and mouse, input. But, currently, it is a very expensive technology. Much of its functionality could be provided at less cost by pen-based microcomputers or portable computers with stylus pads if they were all networked together and running a conferenced whiteboard application. Hand-drawn input could selectively come from any of the computers, and output from the conferenced window could be displayed on each computer as well as on a large screen using inexpensive overhead projection equipment. The missing function would be direct drawing on the projected display, but it is hard to see this as a significant limitation if each participant could draw on his or her personal electronic tablet and have the result shown on the central display. This configuration would also provide greater continuity between collective work that takes place in meetings and individual work that comes before and after.

Although only beginning to appear on the commercial market, meeting support tools could produce significant changes in the ways groups work by producing a change in the forms of information they work with in those situations. In a conventional meeting, most of the interaction is verbal. Usually there is no literal record of a meeting, and participants often disagree about what actually took place. To help them iron out differences in their respective understandings of what they are talking about, groups often use ephemeral products — such as whiteboard drawings, foils, video, and so forth — that serve in the moment but are not kept permanently. Even if a meeting focuses on some part of the artifact, as would occur in a design review or group editing session, the actual tangible product discussed is ephemeral — for example, printouts of code or documents maintained

in a computer system rather than the actual artifact. After the meeting, someone must update the permanent version, accordingly. Because the literal product the group worked on is not preserved, this can lead to disagreements about what was actually agreed to. Thus, most meetings are concerned with intangible or ephemeral forms of information.

In a meeting that includes a meeting support system, products that would normally be ephemeral, and hence lost, could become tangible products that are stored permanently as part of the artifact. For example, whiteboard drawings that are normally erased could be stored in computer files if they were produced on a Liveboard rather than a whiteboard. During a meeting, the group could also address the artifact directly, rather than ephemeral products derived from it. Thus, groups using meeting support systems should show a significant shift toward more tangible forms of information and more focused discussion.

It would seem obvious that a change that makes discussion more tangible would be beneficial, but this may not always be true. For example, during a series of brainstorming sessions, such a shift could lead to premature closure around an early position that is retained in tangible form and repeatedly brought back into the discussion, whereas a conventional, less tangible discussion might have led to consideration of a wider range of options. Additional research is needed to understand the many subtle ways these tools will affect collaborative behavior and to help groups make effective use of them.

A Comprehensive System

In the two preceding sections, I described tools that support the independent, asynchronous work of groups as well as tools that support their real-time, synchronous activities. Groups, however, do not make clean distinctions between these two modes of work. For example, an individual working alone may call a colleague into his or her office to ask that person's opinion about a paragraph he or she is working on. Or, a question addressed to one individual across a computer lab can lead to an informal discussion joined by several other people working nearby that, in turn, leads to a search for a

particular piece of code that the group then discusses. If each activity is supported by a different system, shifting from one to another may be awkward, and moving data from one context to another may be difficult or impossible. Instead, groups need systems that span the spectrum of synchronous and asynchronous activities and permit quick, easy transitions from one mode of work to another. More specifically, they need comprehensive systems that combine most of the individual tools described previously and can accommodate new ones not yet developed.

Over the past several years, the UNC Collaboratory Project has built a system that is trying to meet this need. Called the Artifact-Based Collaboration (ABC) system, it supports development of the artifact by groups engaged in both synchronous and asynchronous work (Smith & Smith, 1991). It also includes tools for creating and using ephemeral products and for developing shared intangible knowledge. Thus, it supports the three basic types of information included in the information flow model, although not necessarily all situations in which each occurs.

In Part II, it will be easier to discuss collective intelligence as a form of computer-based behavior if we can assume that groups have access to a particular set of tools. Consequently, I use ABC as the example system that provides that basic set. However, ABC is not the only system that supports multiple activities; others include Doug Engelbart's Augment (Engelbart et al., 1973), whose legacy underlies virtually all work in this field, Mermaid (Watabe, Sakata, Maeno, Fukuoka, & Maebara, 1990), SEPIA (Streitz et al., 1991), and HB1 (Schnase, Leggett, & Hicks, 1991).

Overview of ABC

Figure 3.1 provides an overview of ABC. The system is designed to be used by distributed groups whose members may be located a considerable distances from one another, clustered at a single site, or combinations of clusters and scattered individuals. Thus, a key assumption is that members do much of their work through workstations connected to a high-speed network. This network connects each workstation with a *hypermedia data storage* facility as well as with other workstations. Users work with *browsers* that

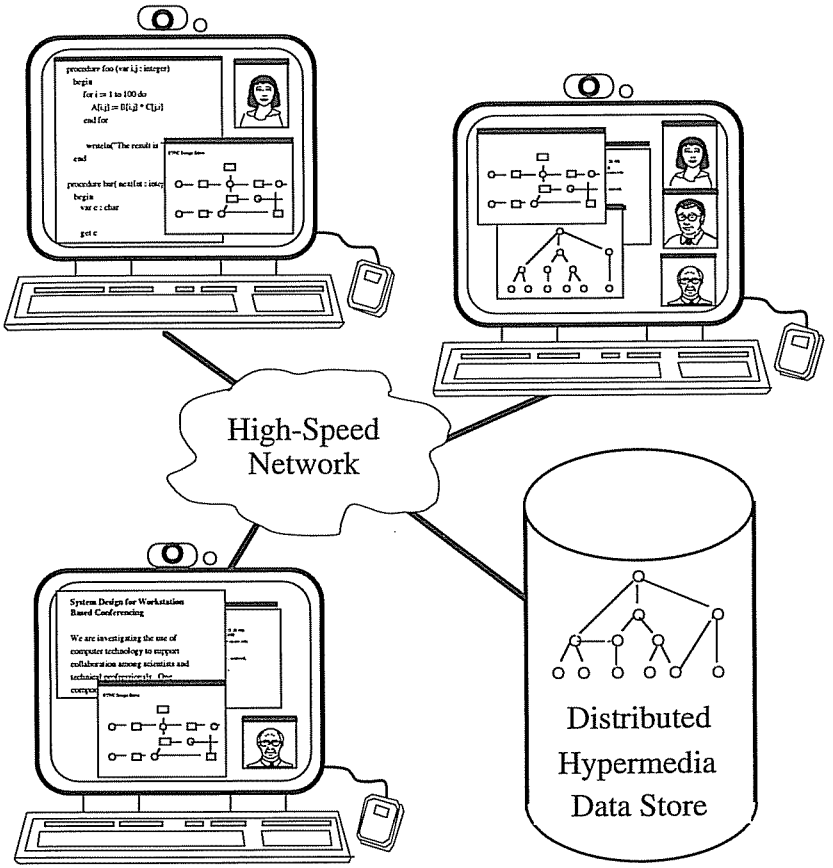

Fig. 3.1. Overview of the Artifact-Based Collaboration (ABC) System. Three workstations are connected to a hypermedia storage system and to one another by a high-speed network. The same conferenced browser can be seen on each workstation, with supporting video windows. Other nonconferenced browsers and applications can also be seen on each workstation.

enable them to see and manipulate structural data and with *applications* that enable them to work with traditional "content" data, such as text and diagrams. A *conferencing* component allows several members to share any browser or application so that all can see the same information and each, in turn, can edit or manipulate the data. The design also includes *audio and video* communication so that members

can talk to and see one another, both to supplement computer conferences and for conventional conversations. Thus, ABC integrates support for three major forms of collaborative work — individual work on the artifact, collective work on the artifact, and conversations/discussion — thereby supporting both synchronous and asynchronous activities.

To understand ABC more clearly, one must understand five key concepts or system features: virtual screen, hypermedia data store, browsers and applications, conferencing, and audio/video communication. Each of these is discussed later.

Virtual Screen

ABC runs within the X Window System under the UNIX operating system. However, the data users work with is stored in the hypermedia data storage system, described later, rather than the UNIX file system. This requires users of ABC to have a different mental model of their data and to use different tools to work with it. In addition, the system provides several kinds of generic function, such as conferencing and hyperlinking, that apply to any program or application running within it. Consequently, it is important for users to be aware of whether they are working within UNIX or within ABC at any given moment.

To do this, ABC provides a facility called a virtual screen (Jeffay, Lin, Menges, Smith, & Smith, 1992). It is a window with an identifying label. However, it can include other windows, and it can be expanded to cover the entire screen. All ABC programs run within this virtual screen, and any program that runs within it references the hypermedia data store, rather than the UNIX file system. Access to ABC generic functions is provided by an additional menu bar attached to the top of each program window within the ABC virtual screen.

Thus, ABC can be viewed as an environment within the larger UNIX context, or it can be expanded to appear to be the entire system.

Hypermedia Data Store

ABC encourages a group to think of the entire collection of information it builds and works with as a single, integrated structure, rather than as a set of separate and independent files (Shackelford,

Smith, & Smith, 1993). This structure constitutes the group's artifact. For a software development project, the artifact might include early concept papers, requirements, specifications, the architecture, source code, maintenance manuals, user documentation, and perhaps conference papers and journal articles that describe the system. It could also include the private or personal data of the individual members of the team.

In terms of the ABC data model, the artifact consists of a collection of separate *graphs* that are composed to form a single large, interrelated structure. Currently, ABC supports three types of graphs: *trees, networks,* and *lists*. Each separate graph corresponds to some presumably logical entity, such as a short document or a section or chapter of a larger document. Graphs, in turn, consist of sets of *nodes* and *links*. Nodes normally represent concepts, whereas links represent relationships between concepts.

Some graph types are better suited for particular tasks than others. For example, one could use a tree to represent the hierarchical structure of a document, similar to an outline. The title would appear as the label of the node at the top, nodes for major sections under that, subsection nodes below each section, and so forth. However, because tree graphs permit only links from a "parent" node to its "child" nodes, they do not allow links between "sibling" nodes or links that would otherwise cross the hierarchy. If more flexibility is needed, for example, to record ideas and relationships generated during a brainstorming session, one can use a network graph, which permits links between any pair of nodes.

An important concept in ABC is node *content*. Although nodes can be used to represent concepts directly, they can also contain two types of information. First, nodes can contain a block of data similar to a conventional file. For a short document represented as a tree, the actual text, diagrams, or other forms of information that constitute its substantive content can be stored as the contents of the "leaf" nodes at the bottom of a tree. A printed version of the document can, then, be derived by having the system go around the tree and gather up all of the pieces stored as node contents and send them to a printer. Second, nodes can contain graphs. That is, separate, disjoint graphs can be stored as the contents of nodes, just like the blocks of data described previously. For example, a group could collectively develop the overall structure of a document as a two-level overview tree that identifies the document as a whole and the chapters to be included in

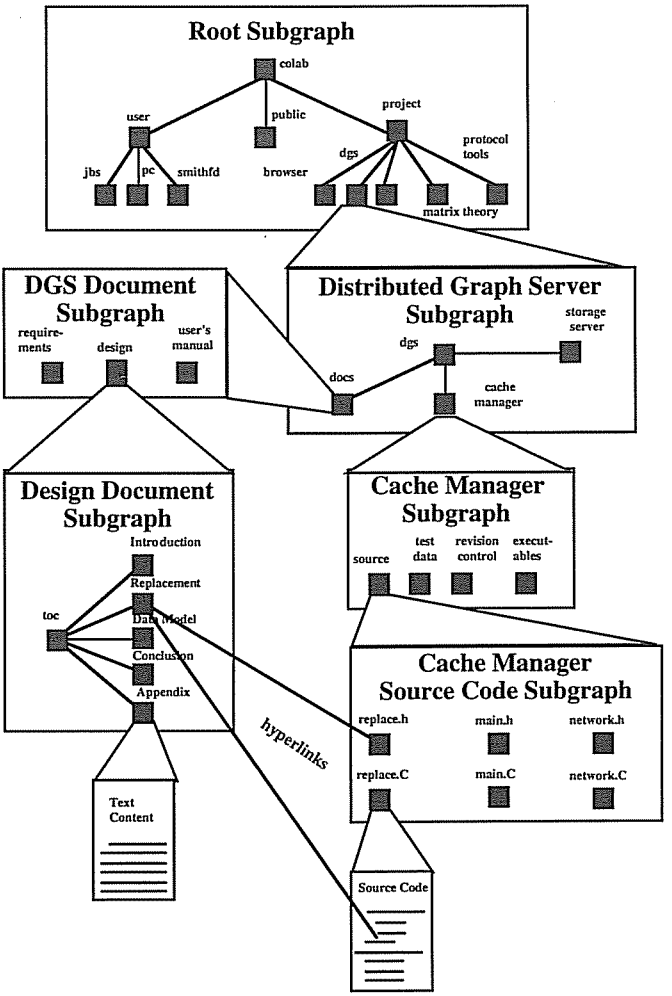

Fig. 3.2. Artifact for a collaborative project. Higher level nodes contain graphs, whereas nodes in lower level graphs contain file data, in this case, text. Hyperlinks connect a node in one graph with nodes in another, one of which is anchored within the node's content.

it. They could then work individually or in teams on the detailed plans for each chapter as separate graphs that are contained within the chapter nodes. Indeed, this pattern of work was seen in several of the scenarios described in chapter 2. By successively deeper nesting of graphs, groups can build structures that are very large, but ones that can be viewed at different levels of detail, making them easier to understand and easier to work with. Figure 3.2 illustrates how a group might organize all of its information as a single artifact.

Finally, the ABC data model includes a second type of link, called a *hyperlink*. The links described to this point are *structural* links. They are constrained by the rules that apply to a given graph type. For example, a node in a tree graph can have (at most) one incoming link, but any number of outgoing links. This constraint, although helpful in some contexts, precludes relationships that are also desirable, such as cross references, that would violate the basic tree structure. Hyperlinks provide this flexibility without violating the integrity of the graph type.

Hyperlinks can define two types of relationships that cannot be defined using structural links. First, they can join nodes (or points within the contents of nodes) to one another within a graph that would violate the type of the graph if the relationship were defined using a structural link. Second, hyperlinks can join nodes in one graph to nodes in another graph. Because a structural link must exist within some particular graph, they cannot be used for this purpose. This second use of hyperlinks is especially important for collaborative work. As discussed in chapter 2, maintaining the internal consistency of the artifact becomes a major problem for large projects. Hyperlinks provide a mechanism through which groups can identify dependencies within the artifact so that when a change is made in one place — for example, a specification document — one can follow hyperlinks to other places within the artifact — for example, the source code and user documentation — that may be affected by the change and, thus, require update or verification. Figure 3.2 shows both types of hyperlinks.

Browsers and Applications

Members of a collaborative group work with ABC through two types of tools: browsers and applications. Browsers enable users to

view and work with the structure of the artifact. Through them, users create new graphs as the contents of nodes; add, delete, and move the nodes within a graph; create and edit links between nodes; add or change identifying labels on nodes; or simply view an existing graph. Currently, ABC includes browsers for trees, networks, and lists, corresponding to the three types of graphs supported by the data model. Other browsers could be developed, such as a decomposition diagram editor, whose underlying data model is a graph but whose appearance differs from conventional graph representations.

Applications are used to work with blocks of nongraph data. Examples include text editors, drawing programs, spreadsheets, CAD/CAM tools, and so forth. ABC provides an open architecture with respect to applications, so long as the application operates within the X/UNIX environment. Thus, a group can use ABC browsers for working with the structure of the artifact, but the tools it is accustomed to using for working with the individual files/blocks of data contained within the graph structure.

Conferencing

ABC supports shared X conferencing as one of the generic functions it adds to any browser or application running within it. Conferencing is accessed through the second title bar ABC adds to a virtual screen and to all browsers and applications that appear within it (Jeffay, Lin, Menges, Smith, & Smith, 1992). This allows members of a distributed group to work together at the same time on a document, drawing, or other form of information by sharing any browser and application running in the ABC environment. They all see the same display, and they can take turns providing input to the conferenced program.

One unusual feature of ABC conferencing is that it permits multiple browsers/applications to be shared within a single conference. This can be done by conferencing a virtual screen and all browsers and applications running within it, as opposed to a single program. Thus, for example, if a question comes up during a conference that cannot be answered by reference to currently shared data, a member can start a browser or application on other data and then bring the second program into the conference.

By integrating conferencing into the collaborative environment, ABC enables members of a group to shift easily and smoothly between individual, asynchronous work to synchronous, collective work. And, of course, there is no issue of moving data from one context to the other, because the group works with portions of the artifact in both situations.

Audio and Video

Although computer conferencing can help distributed groups work together on the artifact in ways that would not be possible otherwise, members also need to talk with one another, hold meetings, and carry on both formal and informal discussions. No technology can replace face-to-face encounters. But it may be able to supplement those encounters, particularly for groups that are widely distributed and for whom frequent face-to-face interaction is not possible.

ABC includes audio and video communication within its basic design. Currently, this function is implemented outside the ABC environment proper, using the telephone system for audio and cable television technology for video. In the next stage of our work, we plan to fully integrate audio and video communication into ABC. To do this, we will build on work done by my colleague, Kevin Jeffay (Jeffay, Stone, & Smith, 1992, 1994). Using specialized scheduling algorithms developed by his multimedia research group and conventional data compression hardware, Jeffay's system can deliver full-motion video and high-quality audio over conventional packet-switched data networks. We plan to incorporate this type of digital audio and video as a generic function, analogous to conferencing, that can be accessed from within ABC and controlled by ABC applications. Thus, users will be able to schedule and setup audio/video conversations analogous to starting a computer conference, enabling them to make smooth transitions from individual work, to conversations with colleagues, to video supplemented computer conferences.

Thus, ABC includes a broad range of functions that are normally found in separate collaboration tools. These functions address all of the types of information included in the information flow model as well as key transformations. ABC also supports both synchronous and

asynchronous activities, and the system permits easy transitions from one mode of work to the other.

For asynchronous work, ABC allows distributed groups to work within a single, integrated structure of information — the artifact — with the sense that data is omnipresent, rather than a scattered collection of files that must be moved from one location to another for access. It includes basic facilities with which to note dependencies and other relationships within the artifact that must be maintained if the structure is to be coherent and internally consistent. And its data model provides a natural means of partitioning the artifact for ease of understanding, for supporting multiple concurrent users, and for scalability.

For synchronous work, ABC includes computer conferencing as a generic function that can be applied to any browser or application. When fully implemented, ABC will also provide audio and video communication over conventional data networks. Currently, ABC does not include explicit meeting room support; however, whiteboard applications and other similar meeting tools could be run as ABC applications and used collectively during meetings. In the future, we plan to explore using ABC with new input/output technologies, such as stylus pads, pen-based computers, and liveboards.

Groups have worked in the past and, no doubt, will in the future using conventional computer and communications systems as well as more basic tools, such as pen and paper or typewriters. However, as systems are developed explicitly to support collaboration, it seems safe to assume that they will play a larger and larger role in collective work. For the remainder of this discussion, I assume that collaborative groups are using ABC or an equivalent system so that I can assume they have access to a specific set of capabilities, thereby making it easier to define collective intelligence as a form of *computer-mediated* behavior. Later, one could go back and factor out ABC-specific features to generalize the concept.

Issues for Research

There are a number of issues and questions concerning collaboration systems that need further research. Here, I briefly discuss several that seem particularly important.

- *What is the relationship between the process of collaboration and systems that support that process?*

We need to look closely at the relationship between the needs of users and the services provided by systems. Which collaborative activities are supported by existing systems? Do those tools help groups work better or more efficiently? Are there other collaborative activities for which no supporting tools exist? As better models of the collaborative process are developed, we should continuously map the activities and the information transforming processes identified in those models against supporting tools, both for purposes of evaluation and for identifying gaps where new tools might be built.

- *What should the user interface for collaboration tools be like?*

Should the user interface for a collaboration system be different from the interface for a system intended for individual use? Currently, many systems base their user interface on the metaphor of the desktop. Because a desk is normally used by an individual, is the desktop an appropriate metaphor for collaborative systems, or is there some other metaphor that would be better? Do we really need such a metaphor? The relationship between everyday objects and computer functions quickly breaks down; consequently, metaphors are most helpful for novice users and can become counterproductive for advanced users. We may be better off identifying a common set of user functions needed by most collaborative applications and then presenting them to the user though some consistent, abstract, but nonmetaphoric image, such as a graph.

- *Which facilities can help groups organize and access information in wide area storage systems?*

Which form of wide area storage system is most appropriate for distributed collaborative groups: wide area file systems, distributed database management systems, network accessible document management systems, or closed architecture hypermedia systems? Do different groups need different storage systems, or will one type dominate in the long run? If wide area file systems become predominate, they will enable users to access large, widely distributed, and, most likely, unstructured collections of files; can and should these collections be given more coherent form? If so, is a hypermedia graph model appropriate, or is some other structure better? Should multiple groups be able to construct multiple structures over the same

files? What new problems of consistency and security would this raise? Is replication an essential requirement for performance and, if so, how frequently should replicated data be updated? What sort of browsing and search functions are needed? Does the user need to visualize the information structures he or she works with and, if so, what tools can help?

- *What kinds of interaction are needed among computer applications and processes?*

What should the system and communications infrastructures provide to support shared editing and other forms of real-time application-based interaction? Should support be provided at the level of the window manager or at some deeper level of the system? Are existing protocols, such as X Window, adequate, or is a new protocol needed that assumes collaborative interaction as a basic requirement? What is the proper level of granularity of interaction? Should granularity be a developer or end-user option? How frequently should data be updated? Should all users see the same view (i.e., WYSIWIS: what-you-see-is-what-I-see), or do users need independent views? Are there other conceptual models for real-time application-based interaction in addition to shared editing?

- *How can human and data communications best be combined?*

How can human verbal and visual interaction best be supported among distributed groups? Should support for voice and video be treated as separate services, or should they be integrated into the workstation so that they can be used in conjunction with traditional data applications? Should the phone company or should data networks such as the Internet provide these services? That is, should support for voice and video be developed within data communications protocols and architectures, or should the emphasis be placed on developing better interfaces between computer and telephonic systems? Because most data networks are implemented using leased telephone lines and, conversely, because most telephone systems are digitally based, could dual purpose networks be achieved through standards? If it turns out that human interaction is best supported through data networks, what changes in computer and communications architectures will be needed to address fundamental problems of latency, synchronization, and reliability?

- *What would future systems infrastructure look like if support for collaboration and cooperative work became a basic requirement?*

Virtually all existing computer systems were designed to support individual users working alone. This is obviously true of personal computers and individual workstations. But it is no less true of large, mainframe systems that provide their numerous users with the illusion that each is the sole user of the system (e.g., CMS' concept of the virtual system). Early collaboration systems were treated as applications, supported by this infrastructure. More recently, a few have taken a more general approach by adding architectural layers to the infrastructure, such as a graph abstraction layer on top of distributed file systems or shared application support on top of window management protocols, with the result that they can provide collaboration support for open-ended sets of applications. But, these more general systems are often forced to compromise or to solve problems in ad hoc ways, because they rely on existing operating, file, window manager, and communications systems, none of which were designed with the needs of highly interactive, widely distributed collaborative groups in mind.

What if we began with a blank sheet of paper? What fundamental changes would result in system and communications architectures if we assumed that support for collaboration and communication among users was a *fundamental* requirement, on the same level as providing permanent storage, allocating resources, managing main memory, providing interprocess communication, and so forth? What would *that* infrastructure look like? Once conceived, could it be implemented on top of existing systems and communications infrastructures, or would it require building from the ground up? If it turns out that the second alternative is needed, could this be done given the enormous legacy of existing hardware and software?