Chapter 6

Collective Processing

A second major component included in the cognitive models and architectures discussed in chapter 4 is a processor. Consequently, we should expect a concept of collective intelligence to also include a *collective processor*. Just as conventional conceptual processing is done in close conjunction with the human memory system, so collective processing is closely related to the collective memory systems discussed in chapter 5.

The original Newell and Simon IPS model included a separate processor component as part of the basic architecture. In the Act* and Soar architectures, however, the processor has been replaced by a working memory component that functions as a cache and provides the context in which a set of processes operate on its contents. Thus, the emphasis has shifted from processor as object to the more abstract concept of a set of processes. I adopt this second perspective, focusing on the different types of processes groups use to get their work done, although for convenience I sometimes refer to these processes as the collective processor.

Before beginning the discussion, I want to point out an important difference in perspective between the IPS view of processor/processes and the collective processor/processes I am describing. Both Newell and Anderson were concerned with fundamental properties of human cognition and problem-solving behavior. Consequently, their models were implemented as autonomous computer systems intended to simulate human mental behavior and, thus, to function independent of the direct control of a human user. These systems are within the mainstream of artificial intelligence research.

By contrast, collective intelligence is situated within a different line of research, called *intelligence amplification*. This perspective emphasizes use of the computer to amplify or supplement human mental capabilities. Thus, a human user remains in control of the computer system. Furthermore, it is the human user, not the

computer, that supplies the basic information processing operations required to carry out a task.

Just as the collective memory discussed in chapter 5 was comprised of separate subsystems for tangible and intangible forms of information, the collective processor also includes corresponding sets of processes used to develop these two types of information. A third type of process occurs in situations in which both types of information are simultaneously activated and developed by a group. The discussion that follows examines each of these three types of processes.

Processor for Tangible Knowledge

The conceptual processes used by members of a collaborative group to develop the artifact can be viewed as a *collective processor for tangible knowledge*. It includes operations carried out by individual members of the group working alone as well as the aggregate of their multiple independent activities. In this section, I will begin by focusing on a single individual working alone and then on multiple individuals working concurrently, but independently, within a group.

Individual Processor

Although the overall collaborative process includes situations in which the group works together in the same room at the same time, for many projects, the majority of time is spent in individual work in which members work independently at their respective desks and/or workstations. In chapter 2, I referred to this second mode of work as asynchronous collaboration. Consequently, a concept of collective intelligence must account for individual, asynchronous work on the collective enterprise as well as the group's more interrelated activities.

In recent cognitive models and architectures, working memory provides the context in which processes are applied to activated portions of long-term memory or to new input data. In chapter 5, I argued that the artifact, maintained as a hypermedia graph structure,

can be viewed as a form of long-term memory for tangible information. Similarly, I showed that the various browsers and applications included in a collaboration support system can be viewed as forms of working memory. Consequently, the processes that groups use to develop tangible knowledge are closely related to the tools they have for working with the artifact.

In the context of a collaboration support system such as ABC, it is the individual user who supplies the basic mental processes responsible for representing new concepts within a browser, for perceiving and denoting new relationships between existing concepts, for building and modifying larger conceptual structures, for saving the results in the computer system's long-term storage facility, and for carrying out other information processing operations related to the artifact. The relationship between user as source of conceptual processes and the information to which those processes are applied is suggested in Fig. 6.1.

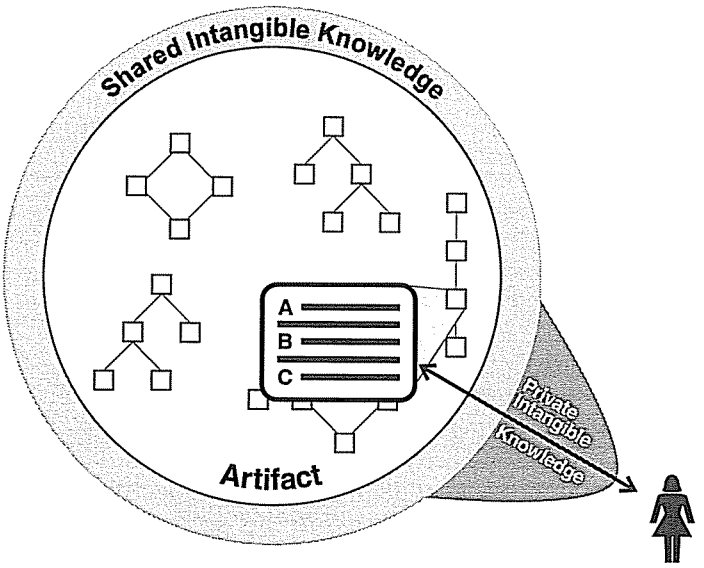

Fig. 6.1. Individual processor, working with an activated portion of the artifact, informed by both private and shared intangible knowledge.

In the figure, the tangible artifact is shown in the center. A part of it has been activated within a browser or application. The user, who, of course, is external to the computer system, is attending to that activated segment. As the user performs various operations on this information, he or she does so through a perspective shaped by his or her intangible knowledge. This includes knowledge shared with other members of the group, suggested by the lightly shaded area that surrounds the artifact, and knowledge held privately by the particular individual, such as specific technical expertise, indicated by the darker, uneven areas that surround the shared knowledge. Each individual member is associated with a different body of private intangible knowledge.

Work progresses through a sequence of changes made to the artifact. The basic unit of conceptual work involves a *cycle* of interaction between human user and supporting computer system. The primary function of this cycle is to keep the representation of information within the computer system consistent with the user's evolving mental state. Consequently, I call this basic cycle *computer-mediated cognition* (cmc) to emphasize the mediating effect the computer has on the user's thinking and to distinguish it from mental operations that take place without direct reference to representations maintained in such a system. The general form of this cycle is suggested in Fig. 6.2.

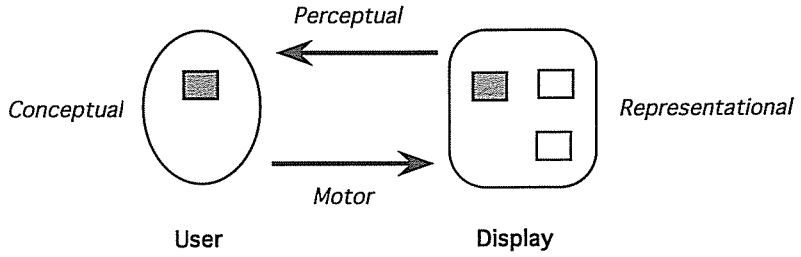

Fig. 6.2. Computer-mediated cognition cycles involve cycles of conceptual, motor, representational, and perceptual operations. The object of current attention is indicated by the shaded box.

The user interacts with the system through an ongoing sequence of conceptual, motor, representational, and perceptual operations.

The user's mental state may change as a result of some conceptual process independent of the current state of the computer system. When this occurs, he or she may elect to represent the change in the computer through a series of motor actions that control the system, such as moving a mouse or typing on a keyboard. When those actions are completed, the user perceives the change in the display, completing the cycle. In other cases, the conceptual process may be triggered by the representation, such as the user seeing that two concepts currently shown as independent of one another can be grouped to form a conceptual cluster. Again, a series of motor actions can be used to update the display in order to make it consistent with the user's new understanding of the data. Different forms of this basic computer-mediated cognition cycle are associated with different conceptual processes and produce different kinds of changes in the artifact.

To examine this concept of cmc cycles more closely, let's look at an example — the cycle used to add a new concept to the artifact — shown in Fig. 6.3. It describes the process by which a group member retrieves a concept from his or her (human) long-term memory and then represents it within the computer system. The cycle begins with a conventional memory access that produces a new conceptual object in the user's conventional working memory. I refer to this new concept as a "delta product" to indicate that it constitutes a change in the set of concepts currently available in the user's working memory for consideration; hence, it is shown in the figure as a "C" within a triangle or "delta" symbol. This process is shown in the first line of the figure.

Once the concept becomes available for attention, it is subjected to several tests, some of which may be unconscious or automatic. These tests — not necessarily in the order shown in the figure — determine whether or not the user regards the new concept as relevant to the current semantic context, whether or not it is worth representing, and whether or not it is already represented in the display. If any of these tests fails, this particular cmc cycle is terminated. If the concept passes these tests, a goal is generated by the user to represent it in the computer system.

To accomplish this goal, the user again accesses his or her long-term memory to retrieve a method by which to do so. It is comprised of knowledge of how the computer system works, consisting of one or more computer operations he or she knows how to use to create new objects in the display. Activating the method results in a sequence of motor actions performed by the user, interspersed with perceptual acts

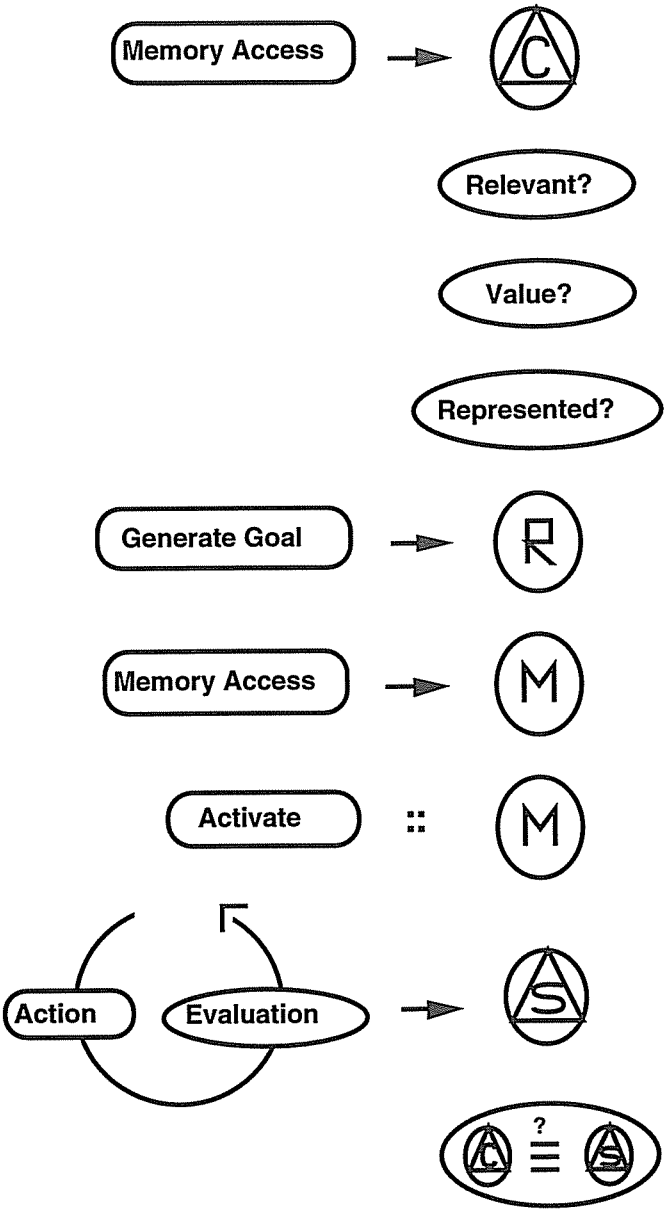

Fig. 6.3. Define_Concept computer-mediated cognition cycle.

to adjust those motor actions. For example, in an ABC network browser, the user points with the mouse to a position in the window, selects the *create node* option from a menu, types a word or phrase to denote the concept, and then types a carriage return to complete the operation. Successful completion of the method, thus, produces a new object within the computer system that corresponds to the conceptual object in the working memory of the user.

Once this new computer system object exists, it is subjected to one final test — to confirm that it is a satisfactory approximation of the original concept in the user's mind. If it is satisfactory, the cycle ends and another one begins; if it is not, an editing cycle is normally initiated (although the user could decide to live with the "unsatisfactory" representation).

Each such cycle constitutes one basic process within the overall set of processes that constitute the collective processor. Examples of other processes/cycles include relating two or more concepts to one another, adding a concept to a cluster or category of concepts, adding or deleting concepts to larger conceptual structures, and translating an abstract idea into prose or other form of expression. Complex knowledge-construction tasks require a number of different processes. For example, the task model for expository writing shown in Fig. 4.9 includes 21 different processes that occur in seven different cognitive modes. Different tasks require different sets of processes; however, at this very basic level, many tasks overlap. For example, many include one or more planning modes and, thus, share some of the same computer-mediated cognition processes used for abstract planning. However, tasks devolve into their respective idioms of expression, such as words, computer code, or technical diagrams, as abstract design is translated into concrete terms. Thus, we might speculate that the overall collaborative process will include tens, rather than hundreds, of basic cmc processes and a smaller number of more specialized cycles for particular tasks. A goal for future research is to identify the specific cmc cycles that comprise these collections.

In chapter 4, I suggested that under some circumstances we could consider the human user and supporting computer system a form of abstract, composite human–computer system. Computer-mediated cognition cycles can help us define this notion more precisely. Cmc cycles begin with a change in the conceptual structure(s) within a user's working memory, produced by some cognitive process. They also include a method through which the user updates the data in the computer system to make the representation consistent with his or her

mental state. Thus, each cycle binds a cognitive process to a computer operation (or short sequence of operations that are used conventionally). It is this binding that allows us to think of the computer as an extension of the user's working memory, comparable to other forms of extended memory but more dynamic than most. Conversely, it lets us think of the user as supplying the essential information processing operations that change the artifact stored in the computer's storage system, which functions as a long-term memory for the group's tangible knowledge.

Ideally, the two subsystems should be closely tuned to one another. One way of doing this is to match the design of the computer's user interface and the functions it supports to the user's task model so that the user does not have to translate extensively between the two or to perform unproductive cognitive operations simply to control the computer. When this is the case and when the computer extends basic user capabilities, such as the number of concepts that can be attended to, it can be said to amplify the user's mental capabilities and, thus, his or her intelligence.

We can now define a general architecture for an intelligence amplifying human–computer system. It includes:

- a task model, expressed as a set of cognitive modes and the paths along which information flows from one mode to another
- a computer system that includes a corresponding set of interface modes and data transfer mechanisms between them
- a set of cmc cycles that bind task model to system model
- a human being who performs the task by using the system and engaging the cognitive modes identified in the task model

To be considered a true IA system, the task model must accurately reflect the user's mental habits. Thus, more than one system design may be required if different user populations perform the task differently, or, alternatively, the system must be adaptable to different users' strategies. This second approach may be preferable, since it would also allow the computer system to evolve along with users' behaviors as they become more proficient at both performing the task and using the computer system.

In summary, the processes that produce changes in the group's body of intangible knowledge can be described in terms of a set of

computer-mediated cognition cycles. These cycles link a human user, who is the source of conceptual processes, with the components within a collaboration support system, such as ABC, that function as forms of working and long-term memory. This configuration is recognizable as an extrapolation of conventional cognitive models and architectures. Finally, when system model closely resembles users' task model, the system can provide a form of intelligence amplification.

Multiple Independent Processors

The processor described in the preceding section consisted of a single source of processing operations provided by a single human user of a collaboration support system. In this section, I discuss a collective processor that applies to the multiple individuals that comprise a collaborative group. The critical issue is to identify a concept of processor that can account for the independent activities of individual members yet also show how those various activities meld into a coherent whole that represents the overall behavior of the group. The discussion is limited to the processing of tangible knowledge, and I again assume that groups developing this form of information are working with a collaboration support system, such as ABC.

At least two notions of multiple processors are possible. First, the individual members could take turns functioning as the single processor described previously. One member would use the system at a time to work on the artifact. This would enable the group to utilize the specialized knowledge and expertise of its different members, and work could proceed 24 hours a day. However, a strategy or a system that restricted work on the artifact to only one member at a time would defeat many of the purposes for forming a group in the first place. At some point, there would simply not be enough working time to accomplish the task. Consequently, this mode of collaboration and the form of serial multiple processors it implies is not considered further.

A second notion of multiple processors — and the one that is examined in the remainder of this discussion — includes multiple individuals working on different parts of the artifact at the same time, permitting work to proceed in parallel. This view of multiple concurrent processors is illustrated in Fig. 6.4. The artifact is located

in the center. Multiple individuals in the group may work concurrently on different parts of it. The work of each is informed by the intangible knowledge he or she shares with other members, represented by the lightly shaded area, and by his or her private knowledge, represented by the darker shaded areas. To simplify the drawing, the working memory of each users is not shown but presumed as represented in Fig. 6.1.

This mode of work is made possible by systems, such as ABC, that support multiple concurrent users. Members of the group use browsers and applications to activate parts of the artifact, to make changes to those portions, and to save the modified segments. Within the terms of this discussion, each such tool constitutes a separate working memory, whereas the artifact represents the group long-term memory for tangible knowledge. Consequently, we can view each member using one of these tools as an independent conceptual processor, similar to the individual processor discussed in the preceding section. Thus, the multiple individuals who work with multiple instances of the system interface function as multiple conceptual processors.

While each configuration of a user, a working memory, and the group long-term memory resembles the IPS architectures discussed in chapter 4, what about the whole? What form of information processing system is it?

As already noted, this construct, from one perspective, is an extrapolation of the single processor model because it replicates processors and working memories around a core long-term memory, like the spokes of a wheel around a hub. But, unlike a wheel, coordinating the behaviors of these "spokes" is not straightforward. The multiple processors are bound to one another by the collaboration support system, but not rigidly so as the spokes of a wheel are bound to one another by the hub. Rather, they are more loosely coupled and, hence, it is harder to coordinate their activities and to see the coherence of their collective actions. Indeed, it is precisely this independent but coordinated form of behavior that permits a collaborative group to utilize the concurrent efforts of its members. Thus, there is a *fundamental* tension between the dependencies within a group that are ultimately responsible for giving their collective work coherence and the independence of their respective actions that enables work to proceed in parallel.

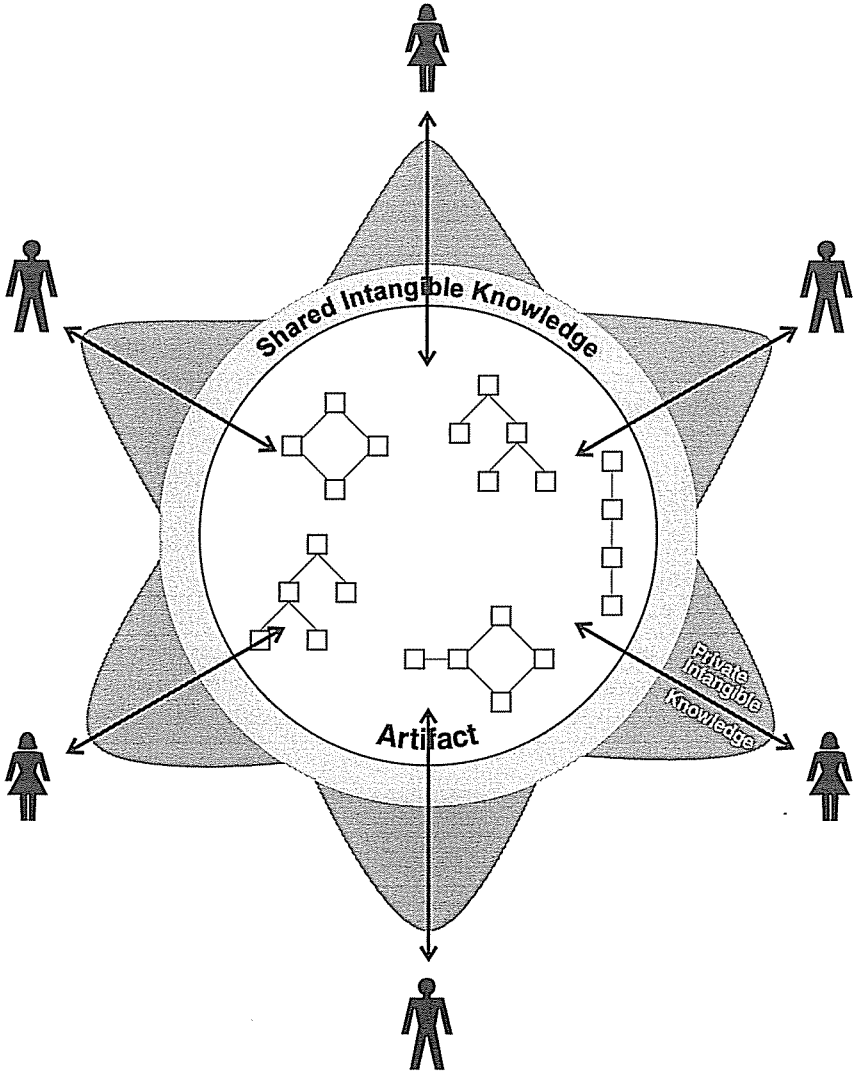

Fig. 6.4. Multiple independent processors form a collective processor for a group's tangible knowledge. Processing is informed by each individual's private and shared intangible knowledge.

To develop a concept of collective processor that includes multiple independent processors but is also coherent, we must extend the basic IPS architecture. This extension can be summarized in a simple proposition, which appears later. However, it will be easier to understand that proposition if we first place it in an historical context.

The original IPS perspective grew out of the predominant architecture for computer systems during the 1950s and 1960s. This so-called Von Neumann architecture, shown in Fig. 6.5, included a single processor, called the central processing unit (CPU), and a single storage component, called the memory. Data were loaded into the CPU from memory for processing and, after processing, results were returned to memory for storage. The CPU was also attached to input devices, such as card readers, and output devices, such as printers. This architecture grew out of the technologies available at the time and contemporary designers' understanding of control, the relationship between program and data, and other similar issues. It is ironic that these early machines were sometimes called "electronic brains" because they bore such little physical resemblance to that human organ.

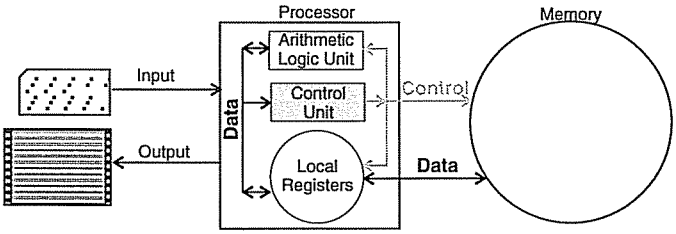

Fig. 6.5. Von Neumann computer architecture.

In spite of these differences, Newell and Simon built their IPS model of the human mind as an extended metaphor with respect to the Von Neumann computer. They abstracted away the physical structure of the machine to identify an underlying architecture that emphasized the storage, flow, and processing of information. That architecture provided a set of terms, components, and relationships in which to build a model of problem-solving behavior. If one compares the basic IPS model, shown in Fig. 4.1, with the Von Neumann architecture, shown in Fig. 6.5, the similarities are obvious. Looking at the human mind as analogous to a computer was, of course, a reductionist view.

But it allowed them to identify information processing as the *essential* activity and to use formal representations, such as computer programs, to describe basic conceptual operations. It also provided a vehicle, in the form of simulation systems, to test their models.

In the interval since the IPS model was first developed, several major new computer architectures have emerged. Those most relevant for this discussion includes multiple processors. They fall into two main families: tightly coupled and loosely coupled systems. Tightly coupled systems are more commonly referred to as parallel processor or multiprocessor systems. We can think of them as a single computer that includes multiple processors within it, each under the control of a central operating system. The processors communicate with one another by sharing a portion of computer memory and/or by exchanging messages through an internal bus. Other defining characteristics include the close proximity of the processors to one another — measured in inches up to several feet — and the system's vulnerability to component failure — when a processor or other component fails, the whole system usually fails (Mullender, 1989).

Loosely coupled systems are commonly referred to as distributed systems. They consist of a number of independent processors connected to one another by a communications network, such as a local area network (LAN); an example distributed system is shown in Fig. 6.6. Each processor is a complete computer, such as a workstation or personal computer. The system may also include specialized processors, such as file servers and high-speed compute servers (e.g., parallel processor computers) that provide services to the workstations or other processors. Each processor runs its own copy of an operating system. Thus, there is no overarching central control in a distributed system. Distributed systems also use shared memory and message passing as means of communication, but they differ from tightly coupled systems in that shared memory may be distributed across the separate workstations, rather than being centrally located, and messages are exchanged over the communications network, rather than over an internal bus. Thus, processors may be located at considerable distances from one another, measured in miles rather than feet. Distributed systems are also designed so that when one element fails, it does not cause the whole system to fail.

If we look at collaborative groups from an information processing perspective, they more closely resemble a loosely coupled

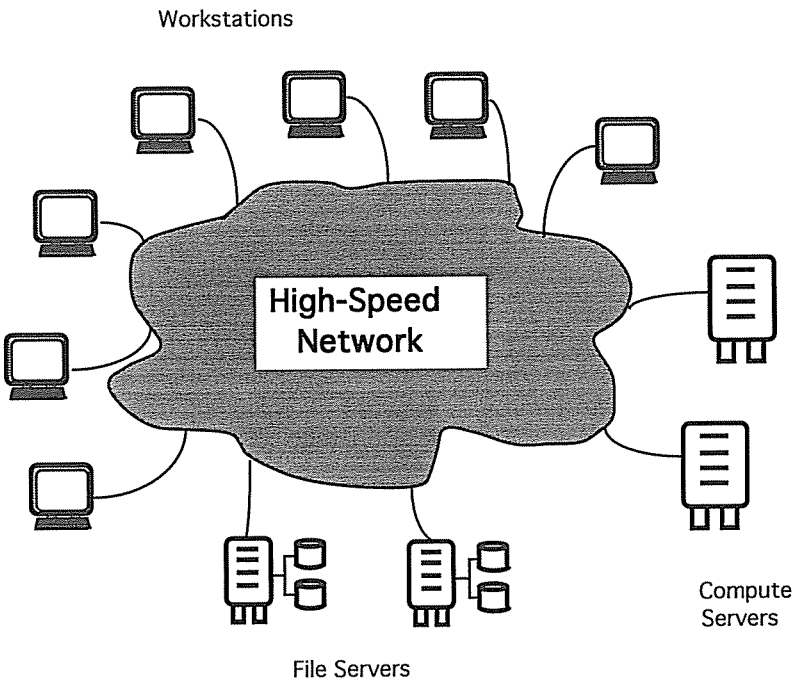

Fig. 6.6. Loosely coupled distributed system.

distributed system than they do either the single processor Von Neumann architecture or a tightly coupled parallel processor architecture. Consequently, if we wish to develop IPS models of collaboration, a better starting point would be the distributed system architecture, rather than the Von Neumann architecture that served as the base for the original IPS models of individual cognition. This proposition can be summarized as follows:

IPS-Indv. : Von Neumann Arch. :: IPS-Collab. : Dist. Sys. Arch.

Translated into words, the proposition asserts that Information Processing Systems models of individual cognition and problem-solving behavior are to the Von Neumann architecture as Information

Processing Systems models of collaboration are to distributed systems architectures.

In effect, this proposition asserts a metaphoric relationship between collaborative groups and distributed systems. We should not confuse metaphor for model, but a metaphor can help us consider collaboration in several ways. We are led to a set of suggestions in the form of a set of issues important in one domain that may be important in the other. A metaphor can also provide terms, relationships, and visual images that can be tried and then retained or discarded, as warranted. And it can provide a framework that can serve as a starting point for the structure of a theory of collaboration. In the remainder of this section, I want to explore this metaphor more closely by looking at several characteristics of distributed systems and then at similar properties for collaborative groups. (For a more thorough discussion of these and other issues associated with distributed systems, see Coulouris & Dollimore, 1988; Mullender, 1989; Mullender, 1993).

Parallel Processing. The core concept of a distributed system is parallel, independent processing. Because each processor is a complete, separate computer running its own copy of an operating system, each is capable of operating autonomously. From a task perspective, this leads to two concepts of asynchronous parallel processing. First, users may work completely independent of one another if each is working on a separate task unrelated to the task any other user is working on. In this case, the only thing logically joining these users and their workstations may be the underlying file system that is part of the computing infrastructure. A second type of parallel processing can occur when two or more workstations are used to work on the same task. This can occur for tasks that can be decomposed into independent operations that can be carried out separately and whose independent results can be merged to form the solution or an intermediate result. Matrix multiplication and decryption are examples of such tasks. The critical issue for this second type of task is finding a valid and effective decomposition.

Parallel processing is also a fundamental part of most forms of collaboration. As I have emphasized, intellectual groups are usually formed because the task is too large and/or too complex to be done by a single individual. Therefore, to meet time requirements, group members must work in parallel. This mode of work may sometimes resemble the separate task form of parallelism described previously — for example, when team members separately write different sections

of a document — although independent work must always be synthesized in a collaborative group if the work is to be coherent. On the other hand, when critical expertise is held by only a few members of a group, it is often necessary to form teams whose members have complementary skills and have them work together closely on the same task. This second mode of work resembles the same task form of parallelism described previously. Thus, finding an effective task decomposition is as critical for collaborative groups as it is for distributed systems. Key issues for research, then, include how groups divide their work into independent steps that can be carried out concurrently, how large these steps should be, and what constitutes a good versus a bad decomposition.

Communication. If a system is composed of multiple independent processors, those processors must be able to communicate with one another if the system is to function coherently. Processors must be able to issue and respond to messages, update a common store of information, and report their current status. In a distributed system, these interactions are carried out using a communications network to which each processor is linked. However, for effective communication, processors must also understand various conventions and protocols. Some of these are rather mechanical, such as formatting information in a particular way for transporting from one location to another. Others are more social — for example, agreeing to acknowledge receiving a message before carrying out the requested operation, analogous to rules of courtesy in carrying on a conversation. Still others have to do with the integrity of the data, such as guaranteeing that the requested operation will produce a particular change or else no change at all to stored data. Thus, communication covers not just the channels over which messages are exchanged, but a complex, multilevel structure of conventions, agreements, and guarantees.

Communication is obviously an extremely important part of the collaborative process. Information flows between members of a group over the same communications network used by distributed systems in the form of e-mail, ftp files, and other forms of electronic communication. But it also flows through a more complex web of human relationships, both formal and informal. This information occurs in a variety of forms, ranging from memos and meetings to chance encounters in the hallway and back-door calls to a friend to get something done. Each of these interactions can serve useful, even necessary functions, and each has its own social conventions.

Important issues for research are documenting the different types of communication that occur within groups — along with the conventions guarantees, etc., that are part of communication — and understanding how they contribute, individually and as a whole, to the work of the group, especially its efforts to construct a coherent artifact.

Fault Tolerance. No physical system is perfect or will remain operational indefinitely. What designers aim for are systems that will fail gracefully. That is, when a component of the system fails, it may inconvenience some users or degrade performance, but it will not result in catastrophic failure of the entire system. A related issue is component replacement. One must be able to replace a defective component or upgrade a component to take advantage of technical advances without affecting the entire system or its overall design.

Groups must also be fault tolerant. They must be able to survive the failure of an individual or team to accomplish an assigned task. Such failures must first be recognized, through monitoring work, checking milestones, or other means, and the group must have the means to correct the problem or replace the component responsible. Similar problems occur when a member leaves the group or becomes incapacitated and must be replaced. A somewhat different problem occurs when a group discovers that it needs expertise not currently available within its membership and must replace a member or add a new member to provide it. This last situation resembles component upgrade. Documenting the mechanisms groups use to provide different forms of fault tolerance is an area for further research.

Transparency. Transparency in a distributed system is concealing aspects of the system from a user so that it appears to be a whole rather than a collection of separate parts. There are a number of different types of transparency (Coulouris & Dollimore, 1988). Access transparency, for example, enables a user to work with files stored on a workstation's local disk and those stored in a remote file server in the same way. Location transparency enables files and computer services to be accessed without having to know where they are physically maintained. Replication transparency hides the fact that multiple copies of an object may exist at different locations to improve access and reliability. Failure transparency allows users to continue work despite the failure of a particular component, such as a server. Scaling transparency allows the system to expand without having to change its structure or the way users work with it.

The analog of transparency in a collaborative group is concealing aspects of the group from an outside observer so that it appears to be a coherent whole rather than a collection of individuals. Perhaps the most important form of transparency relates to the artifact — that it appears to be the work of a single good mind, rather than a number of separate hands, in its integrity and consistency. We might call this property *artifact transparency* to indicate that the work appears seamless. Analogs for other types of transparency can also be recognized in groups. Expertise critical to a group may be replicated in several members to insure its availability and to provide continuity should a key member leave the group. Similarly, when someone outside a group requests something of the group, such as information or an action, he or she should not have to know which individual in the group will answer the request. Perhaps the hardest form of transparency to meet in a group is scale. When a group grows in size, the change is often visible both inside and outside the group. If people are added to a project because the group is behind schedule, the result can often be a slow down rather than an increase in productivity, caused by the overhead in bringing new members up to speed and the additional complexity in communication and coordination that results from a larger group (Brooks, 1975). Inside the group, members may feel that the character of the group has changed, particularly if it goes from being a small, close-knit team to a significantly larger organization with more formal structure and lines of authority. Thus, techniques and designs that permit scaling and other forms of transparency in distributed systems may also be useful in organizing and supporting groups to provide similar properties, such as those discussed here. Testing that possibility could provide an extensive research agenda

Shared State. Shared state refers to a body of current information two or more processors maintain in their local environments about one another or about the system as a whole. The individual components of a distributed system must maintain a certain minimal amount of information about other components in order to recover from component failures. Otherwise, part of the overall state of the system would be lost when a given workstation or server failed (Mullender, 1989). At the other extreme, determining in any very detailed way the overall state of a distributed system is difficult, if not impossible (Birman, 1989). What one would like to have is a global view of a distributed system, such as that of an imaginary ideal observer who could look down on the system as a whole and see into the operations of its individual processors and the communications

network. No such ideal observer can exist in fact. We are forced to observe distributed systems from within, by having processors report their individual states to a monitor processor. However, between the time events are reported by individual processors and the time those messages arrive at the monitor process and it updates its representation of the global state, a host of additional changes can take place in the system. Thus, because of interdependencies and the time required to exchange messages, one can never be sure that the current state of the system as recorded in the monitor reflects the state as it truly exists at that moment.

Maintaining shared state is also both important and difficult in collaborative groups. The body of shared intangible knowledge a group develops can be viewed as a form of shared state, comparable to the type necessary for fault tolerance discussed previously. On the other hand, it is also difficult to ascertain the overall state of a collaborative group in a detailed way, just as it is for distributed systems. We, too, would like to have an imaginary ideal observer who could look down on a group as a whole and make sense at any given moment of all of the various actions taking place within it. But no such observer is possible. Instead, we have to rely on milestones, reporting procedures, and the intelligence and good will of group members to achieve coherence in collaborative work. I look at this issue in more detail in chapters 7 and 8 in discussing collective strategy and collective awareness, respectively.

Parallel processing, communication, fault tolerance, transparency, and shared state are all important properties of distributed systems that have analogs in collaborative groups. There are numerous other correspondences. For example, both distributed systems and collaborative groups must *synchronize* the actions of their various components/members. Both *replicate* important information in multiple locations/in multiple members. Some include *heterogeneous* components — distributed systems, to permit different types of hardware and software to be used together; groups, to have access to different skills and bodies of expert knowledge held by different individuals. And both carry out basic operations on tangible data in series of discrete operations — *transactions*, in distributed systems; computer-mediated processing cycles, in collaborative groups.

By recognizing this rich metaphoric relationship between distributed systems and collaborative groups, we are led to a number

of suggestions for research and for building a theory of collective intelligence. One form these suggestions take is a set of issues that are important in one domain that may be important in the other. They are not the only issues that should be addressed or even necessarily the most important ones, but they are a ready list that should be checked out. A second type of suggestion is a general framework in which to build theory. This framework is the architecture of distributed systems. It may not be the one that, ultimately, is most appropriate for a theory of collaboration, but it can serve as a starting point. As we study groups and build knowledge about parts of the collaborative process, we can replace the analogous component in the framework with the real thing. By the time we are through, it may be that no piece of the original is left, like replacing the boards of a house one by one. But we will have been guided in our enterprise by a shape of the whole.

To get a more specific sense of what an IPS architecture for collaboration based on a distributed systems architecture might be like, consider a collaborative group using the ABC system. ABC, itself, is a distributed system, as is the component that stores and manages the artifact, the Distributed Graph Storage System (DGS). Although logically central, the DGS is implemented as a set of independent processes that run in parallel on multiple workstations and can store data in separate file systems. The browsers and applications that users work with run as independent processes on multiple workstations. Because they can all access the artifact, these programs, or processes, can be said to communicate with one another using a form of shared memory. In ABC's computer-conferencing facility, they also communicate with one another through a form of message passing.

What is missing from this picture are the members of the group. In the preceding section, I described a concept of intelligence amplification in which human user and computer system are so closely bound to one another that the computer can be viewed as an extension of the user's mental apparatus. It is a straightforward extrapolation to consider the multiple members of the group as being similarly bound to a system, such as ABC. Each member of the group can then be viewed as an individual processor (or source of processing operations) that works on the contents of the collective long-term memory, the artifact. This is done by activating portions of the artifact in individual working memory components — the browsers and applications. It is this composite human–computer system, in which a distributed group of individuals work with one another using a distributed collaboration

support system, that we need to describe in order to formulate an Information Processing System model or architecture for collaboration. And, I suggest, it is this composite human–computer system that may achieve collective intelligence.

In this section, I suggested that to build an architecture for collective intelligence, analogous to a cognitive architecture for individual intelligence, we should begin by considering the architecture of loosely coupled distributed systems. In doing so, we inherit a richly suggestive set of issues that can guide and inform research. We also inherit a general framework that can serve as a starting point for such a theory.

To this point, the discussion has focused on the processing and development of tangible knowledge. To understand the overall collaborative process, we must also understand how groups develop intangible knowledge and how that knowledge affects their more tangible work.

Processor for Intangible Knowledge

In chapter 5, I suggested that the intangible knowledge shared by the various members of a collaborative group can be considered a second form of long-term memory. I also suggested that the various situations in which members of a group discuss and develop shared knowledge can be viewed as forms of a collective working memory. Meetings served as the representative example of these situations. In this section, I consider a form of processor that provides the developmental component for creating and using shared intangible knowledge. Because intangible knowledge is stored in the heads of the members and because the situations in which it is developed all involve human participants, we should look to this same group of collaborators as the source of the various processes used to build and maintain intangible knowledge.

Developing the collective intangible long-term memory is, by definition, an activity that involves multiple human beings. It is theoretically possible that one individual could generate the entire

construct, convey it to the group, and the others merely remember it; however, this behavior is neither desirable nor possible in actuality. Thus, I assume that developing the shared intangible long-term memory is a collaborative process in which all members of the group participate.

In situations, such as meetings, in which intangible knowledge is developed, individuals see events from their own individual points of view. However, the situation, itself, inevitably exerts a strong mediating effect on individual cognitive and conceptual processes. That is, the thinking of each individual is inevitably influenced by the thinking of the other members taking part in the activity, even if it is only to disagree. I refer to this situated form of thinking as *group-mediated cognition* (gmc).

Group-mediated cognition takes place within basic cycles of interaction between the individual and the group. Some gmc processes are (almost) entirely intellectual. Others are primarily social. But many, perhaps most, include both conceptual and social dimensions. For example, an idea voiced by a member of the group is evaluated not just on its intellectual merits but also in accord with the listener's assessment of the person voicing it. This merger of intellectual and social processes is one of two fundamental properties of group-mediated cognition.

A second fundamental property is the tension between the individual and the group. More precisely, it is the tension between the conceptual structure that is held in common and, thus, is said to be shared and the slightly different versions of that structure that exist in the individual working memories of the participants. This tension provides both the energy and the developmental operations that drive this form of collective processing. For example, all participants may share the same core structure, but in some minds parts of that core structure may be linked to additional concepts through private associations. If an individual views these associations as inconsequential or idiosyncratic, he or she is likely to remain silent about them. But, if the individual views them as relevant and feels comfortable speaking up, he or she may describe this "new idea" to the group. The other members hear these comments, apply them to the structures in their respective working memories, and thereby change those structures. When this occurs, shared intangible knowledge is extended.

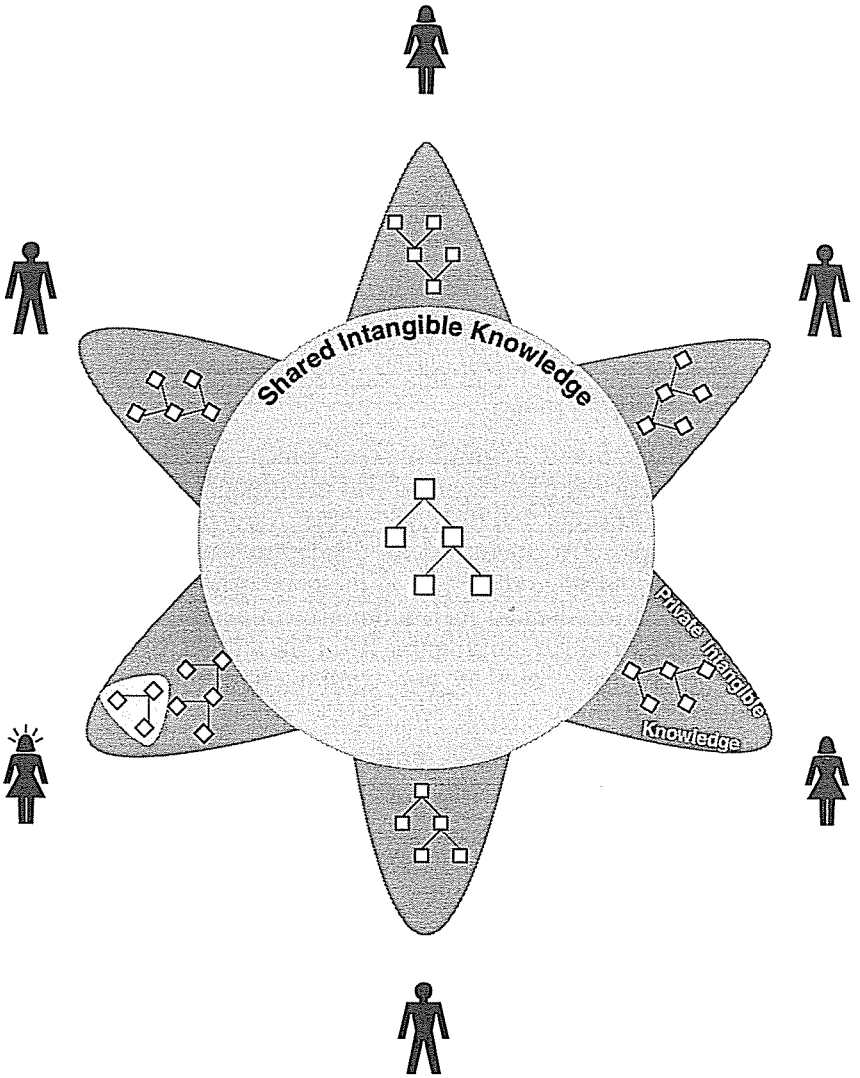

Fig. 6.7. Collective processor for a group's shared intangible knowledge. All participants share the same basic structure of ideas, but one has extended the structure.

The situation just described is portrayed in Fig. 6.7. It provides a context for an *add_concept_to_discussion* gmc cycle that will serve as

the example process that illustrates the concept. A small conceptual structure — such as an earlier decision or a portion of a design — has been brought into the discussion and, thus, activated within the collective working memory of the group. It is shown in the middle of the figure. The individual working memories of six participants are shown at the corners. Although all six versions of the core structure are essentially the same, their orientations are different, suggesting minor differences in perspective. The individual shown at the lower left has made a more basic change in her version by realizing that one of the concepts is related to two additional ideas. She is about to describe the new insight to the group.

Fig. 6.8 shows a sequence of operations for the add_concept_to_discussion process. The cycle begins at the point where the individual has just constructed the extension to a concept included in the core structure. This realization is represented as a change in that individual's conceptual structure and, hence, is identified in the first line of the figure as a "delta concept" operation.

Once the individual is aware of the change, he or she subjects it to two different kinds of tests — one conceptual, the other social. Conceptual tests address the content of the change. Is the change new, or has it already been mentioned by someone else? Is it relevant to the discussion? If so, is it worth mentioning? A parallel set of tests assess social aspects of the situation. Do I feel comfortable speaking? How will the group react to what I may say? Are the potential benefits or losses from speaking worth the risk with regard to my position in the group? These tests are not necessarily performed in the order shown in the figure, and the individual may not be consciously aware that he or she is applying them. One should also note that the two dimensions, conceptual and social, are interdependent, as, for example, seen in the (potential) speaker's assessment of how the group is likely to react to the new concept. If the idea passes these tests, then the individual is likely to make a decision to speak up and describe the new idea to the group.

Once that decision is made, the person will probably spend a few moments formulating a statement. Some people rehearse the actual sentences they plan to say. Others mentally prepare a list of points, but at an abstract level. Still others encode as they speak with little or no prior planning. Regardless of the particular tactic, various planning and/or encoding processes will be used at one or more times to transform the initial change in the core structure into the statement or communication that is subsequently made to the group.

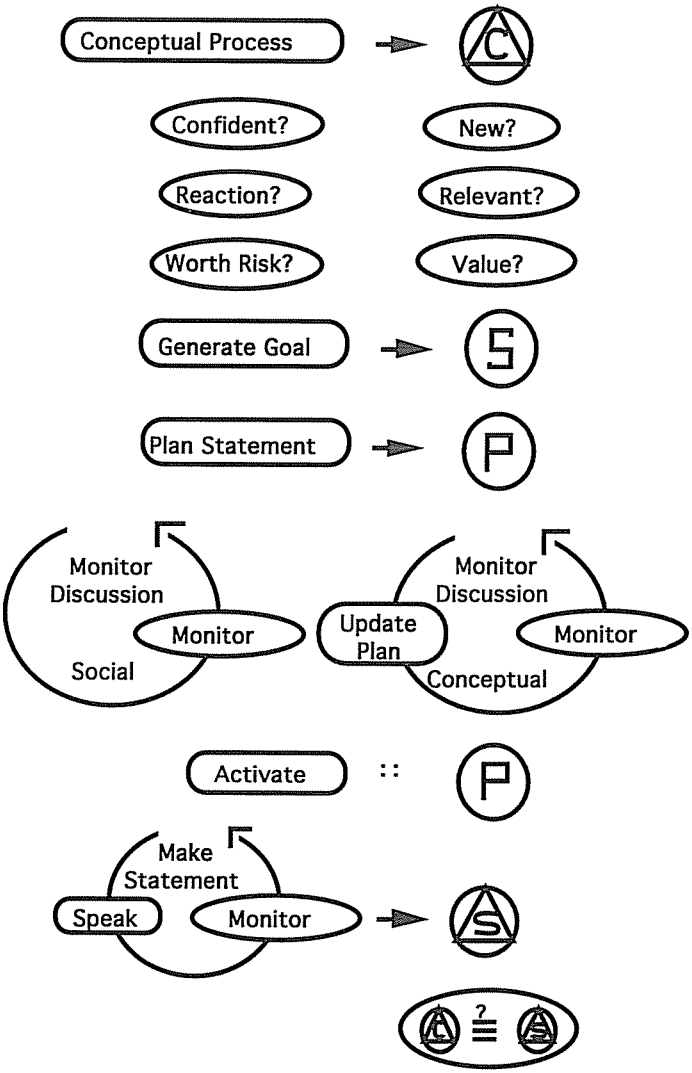

Fig. 6.8. Add_concept_to_discussion group-mediated cognition cycle.

In most meeting situations, people cannot just speak when they are ready. Rather, they must follow some social or organizational protocol to gain the floor. At one extreme, this can be a formal

process of asking the chairperson for recognition. More often, the process is more informal — the individual waits until the current speaker finishes and then speaks up or, if he or she is beaten to the punch by someone else, iterates the monitoring procedure.

During these periods of waiting for a chance to talk, the individual must also monitor the contents of the discussion. Monitoring operations include comprehending the statements made by others, mapping their ideas onto his or her own ideas, and, if need be, updating both the idea the person intends to talk about as well as the plan for the statement itself.

Thus, at least three complex, even contradictory, processes operate during the interval of time between awareness by the individual that he or she has something to say and actually saying it: planning or encoding the statement, monitoring the social or organizational situation to gain the floor, and monitoring the intellectual content of the discussion and reconciling it against the planned statement. Although these processes seem to operate in parallel, further work is needed to determine whether this is so or whether people switch back and forth among them.

Once the floor is gained and the person begins speaking, he or she is likely to monitor the responses of the other members of the group to see if they are following, if they signal agreement or disagreement, and so on. Results of this monitoring procedure can lead to on-the-spot modifications to the statement — for example, the speaker might go into further detail on a point thought not to be understood by the group, or the statement might be cut short if the person feels it is producing a negative reaction.

Once the statement is made, the group may coalesce around it. This may occur spontaneously, or consensus may gradually emerge as influential members voice their agreement. On the other hand, the idea may be ignored as the next speaker begins his or her statement. And, of course, other intermediate scenarios between overt acceptance and rejection are possible. Regardless of the particular scenario, the speaker is likely to monitor these reactions to determine, first, whether or not the other members accurately understood the idea and, second, whether or not the idea is being accepted. If it is accepted, it will probably be remembered and, thus, become part of the core conceptual structure that is shared by the group. In this case, this particular gmc cycle is complete. If not, the individual must decide

whether to pursue the point, to come at it from another angle, or to say nothing and leave the idea to its own fate.

Acceptance by the group, however, is not a requirement for a concept to be remembered by the group. For example, if the group is divided between two points of view on a topic, the discussion may become a debate. If the discussion is memorable and the members share a common understanding of what was said, then the concept can become part of the group's shared knowledge, regardless of whether individuals agree or disagree with it.

Thus, the add_concept cycle is a basic process by which private intangible knowledge becomes shared intangible knowledge. However, it is not the only such operation. Other gmc cycles include establishing new relationships between shared concepts; replacing a concept or a portion of shared knowledge with an alternative concept or structure, through argument or persuasion; translating an abstract concept into some more specific form of expression, such as words, code, or diagrams. Additional research is needed to refine and extend this list of gmc cycles and to articulate the precise sequences of operations that comprise them.

Ephemeral products play a particularly important role in the development of shared intangible knowledge. For knowledge to be shared, the different versions stored in the respective long-term memories of the members must all be approximately the same, but they will not, of course, be identical. Indeed, the developmental process depends upon such variations. The goal, then, is to minimize extraneous differences without constraining the diversity of opinion and individual intellectual contributions that are crucial to the collaborative process.

Ephemeral products help groups achieve a balance between similarity and diversity. For example, during a discussion, when an individual sketches an architecture for a computer system on a whiteboard, the other members can literally see that person's ideas and perspective. Thus, they all share the same image. If that structure is accepted by the group, it is likely to be encoded and retained in their respective long-term memories and, thereby, becomes part of the collective long-term memory. Often, however, the diagram evolves during discussion. For example, someone adds a new component, another changes it to make it interface with another part of the structure, a third re-draws the diagram to simplify it. When this happens, the whiteboard becomes a common field where the group

shares the same structure of ideas and where all members can make their individual contributions. Thus, ephemeral products tend to remove the noise of accidental variations inherent in separate versions of shared knowledge while admitting the free exchange of different points of view.

Finally, I note the similarity between group-mediated cognition cycles and the computer-mediated cognition cycles discussed earlier. Some individual cycles are similar, although not identical, in form, as can be seen by comparing Fig. 6.3 and Fig. 6.8. As a group, the two sets of cycles are similar in function, as the active elements used to develop their respective types of information. A topic for research is to explore similarities and differences between these two types of mediated cognition cycles.

In summary, group-mediated cognition cycles can be viewed as context-sensitive processes that enable a group to develop and maintain a body of shared intangible knowledge. During discussions and other situations where this type of knowledge is developed, participants activate portions of their respective long-term memories, each of which includes both knowledge held in common with the rest of the group as well as private knowledge known only to that individual or to a subset of the group. Once activated, the common structure of ideas evolves through discussion or other similar forms of interaction. The processes responsible for this evolution exhibit two fundamental properties. First, most gmc processes merge social and conceptual operations. Second, the tension between private and shared knowledge is essential for development to take place, although ephemeral products mitigate this tension and mollify extraneous differences. Eventually, a modified version of the shared conceptual structure is encoded and stored in individual long-term memories and, thus, merges back into the collective long-term memory of the group. Thus, group-mediated cognition cycles constitute the basic set of operations that comprise a collective processor for shared intangible knowledge.

Hybrid Processor

In this section, I look briefly at a third type of processor, which I refer to as the *1xn hybrid processor*. It includes multiple processors

operating with respect to a single working memory in which a portion of the artifact has been activated. In these situations, intangible knowledge is also activated and developed.

One form this processor can take is for two or more members of a group to gather around a single workstation to confer about an issue that involves the artifact. They would initiate a browser or application and then either take turns operating the workstation or one member would operate the workstation for the group.

We have observed this behavior in our lab among members of a programming team (Kupstas, 1993). A group of five programmers worked together on a common project in the same room at the same time for some 4 to 6 hours a day over a 3-month period. The workstations were arranged in a "U" with the team members sitting on the inside of the "U". Most of their time was spent in individual work; however, they also interacted with one another at fairly regular intervals, normally for brief periods of time. A common form of interaction was for one member to call over to another to ask for specific information, such as the name of a file or the status of a task. They also frequently "wheeled over" to one another's workstations (they used office-style chairs with wheels) to discuss information displayed on a screen. During episodes of this second type, one member would normally control the system and the other member(s) would view the screen, talk about its contents, sometimes pointing to specific information. Most of the time, the purpose of these encounters was to transfer information — it was easier for the person with the knowledge to show the person than to tell him or her what was wanted. Occasionally, however, they would add new information to the artifact by editing a paragraph or a function, by changing a diagram, and so on. This latter behavior can be considered a hybrid form of processing because the members were both building shared intangible knowledge as well as activating and working with the tangible artifact.

A second form of hybrid processing takes place in computer-supported conferences. Instead of several group members gathering around the same workstation, conferencing systems — several of which were described in chapter 3 — permit participants to work from their respective workstations. Figure 6.9 shows the logical organization of this type of interaction.

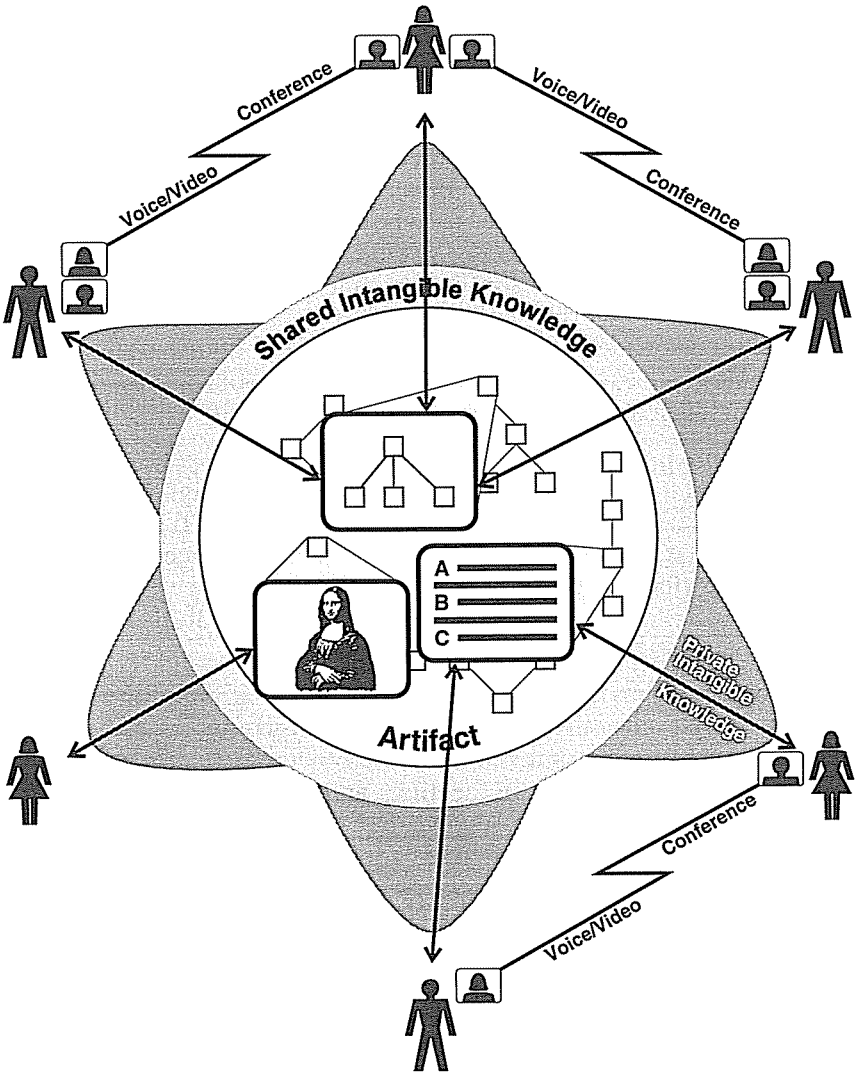

Fig. 6.9. 1xn hybrid processor. Three members are engaged in a computer conference while two others work independently. Conference participants can also see and talk with one another through voice/video communications

When a group works with a conferencing system, one widely used architecture includes a single copy of a browser or application program that runs on a single workstation but selectively takes input from the other participants' workstations and "broadcasts" the program's output to all of their workstations. Although at any one time only a single user normally has control of the system and thus provides input to the program, over a working session, control is normally shared among the participants by passing a symbol, called a *token*, from one user to another — whoever has the token becomes the active user. Some conferencing systems include supplemental voice and video channels that allow participants to talk with and see one another as they work on the artifact. The challenge facing collaboration system builders is to make this distributed form of conferencing as easy and as natural as that of groups working in the same room, such as the group described by Kupstas.

This type of computer conferencing combines aspects of both the processor for tangible knowledge, described in Fig. 6.4, and the processor for intangible knowledge, described in Fig. 6.7. It includes operations that directly affect the artifact and, hence, the group's tangible knowledge. Because participants can also see and talk with one another and, thus, discuss what they are doing, it also resembles the processor for intangible knowledge.

However, the configuration also differs from both of these processors. Because discussion can refer directly to the artifact, it is likely to be more grounded and more objective than conventional discussions that take place in meetings and other situations where the artifact is less accessible. Second, because a computer conference incorporates the private knowledge of its multiple participants as well as the knowledge they share, work on the artifact that takes place in a computer conference is informed by a larger body of intangible knowledge than work performed by an individual member working alone. A topic for research is to document these differences more precisely and under a variety of conditions.

In this section, we have considered two forms of hybrid processor — an assembled form and a distributed form. In the assembled form, a group worked together in the same room at the same time. Interaction was completely natural, and the members moved from individual to collective work easily and instinctively. In the distributed form, behavior is mediated by the technology. Consequently, the ease and naturalness with which members shift between individual and collective work is determined to a great extent

by the design of the support system(s) and current limitations of that technology. These two modes of behavior represent points along a spectrum. However, they are sufficiently alike that we can consider them to be a third type of collective processor. A goal for research is to develop detailed descriptions of this mode of collaboration, including *conference-mediated cognition* cycles that function as the basic processes that operate within it.

In this chapter, I have described three types of processors found in collaborative groups. First, the multiple independent processor supports the individual work of group members as they work on various parts of the artifact. Whereas each member working alone functions as an individual processor — in effect, a single user working with an intelligence amplification system — together they form a multiple processor that resembles a loosely coupled distributed system. Second, the processor for intangible knowledge combines both conceptual and social processes. It, too, includes multiple processors oriented around of common body of shared knowledge. However, because that knowledge is stored in the respective long-term memories of the different members of the group, to say that it is "shared" is only an approximation. Although differences in members' recollections can cause problems, they are also what makes it possible for the group to develop shared knowledge. Third, the hybrid processor combines aspects of both the tangible and intangible processors. Together, these three processors can be viewed as the *collective processor* component of a collective intelligence.

When we attempt to build actual IPS models of collaboration, those models should be based on the architecture of a loosely-coupled distributed system, rather than the Von Neumann architecture that provided the basis for earlier IPS models of individual intelligence.

Issues for Research

Several issues for research are suggested by the preceding discussion. They include the following:

- *Identify a target set of collaborative tasks to be examined.*

If, as a research community, we are to build on one another's work, it is important that we be clear about the domain of study for any given research project. Consequently, as a project begins a study, it should make clear the type of activity and the type of group being examined so that others can know where within the overall space of collaborative behavior the observations and results fit.

- *Identify activities that occur in the target set.*

Once the particular type of collaboration has been determined, as suggested by the preceding issue, a second step is to identify the large-grain behaviors that occur within the group(s) being studied. For individual work, studies of this sort could identify the various cognitive modes used by members to carry out particular tasks. For group work, this would involve identifying the habitual activities the group engages in. Can we identify a "vocabulary" of such behaviors? Is there overlap across groups and tasks with respect to these behaviors?

- *Identify specific mediated cognition cycles.*

The next step is to identify the individual processes that occur within these larger activities. When an activity is carried out by someone working with a computer, basic processes can be associated with a set of computer-mediated cognition cycles. When an activity is carried out in a group situation, such as a meeting, they can be associated with a set of group-mediated cognition cycles. And, when an activity is carried out in a computer-based conference, they can be associated with a set of conference-mediated cognition cycles. Detailed descriptions of the cycles that operate in each of these three processors will provide a fine-grained view of both individual and group behaviors for particular forms of collaboration.

- *Construct an IPS architecture for collective intelligence.*

We need an architecture for collective intelligence analogous to the IPS architecture for individual intelligence. It must include both an overarching framework as well as a set of basic constructs that can be used as building blocks with which to build specific models of collaborations. One promising approach would be to develop this architecture as an analog to the loosely coupled distributed system architecture that includes multiple independent processors coordinated

through shared memory and message passing protocols. This will require a substantial base of prior knowledge about collaborative behavior. But we have a major asset in the examples of Newell, Simon, and Anderson and their development of the original IPS models/architectures. Although the process will not necessarily be the same, we can learn from their experience and methods.