10 Groupware and Computer Supported Cooperative Work

Clarence Ellis and Jacques Wainer

10.1 Introduction

Groupware is hardware and software technology to assist interacting groups. Computer Supported Cooperative Work (CSCW) is the study of how groups work, and how we can implement technology to enhance group interaction and collaboration. This chapter presents definitions, concepts, examples, and issues related to groupware and CSCW. It is written as an overview for the non-specialist, and primarily emphasizes the technical perspective. The material is presented and discussed in the context of a functional 4-part groupware classification. The four categories described within the classification are keepers, coordinators, communicators, and team-agents. This classification is also convenient for the investigation of middleware, and of low level issues of groupware. It also facilitates a discussion of social and organizational implications.

10.1.1 Well-Known Groupware Examples

One way to answer the question of what is groupware is via examples. Groupware comes in many shapes and styles. Most everyone is familiar with electronic mail, and understands that this is a technology used at different times, and different places by its participants. The sender does not expect an immediate reply from the receiver. This can be contrasted with face-to-face electronic meeting room technology, sometimes called group decision support systems (GDSS.) These systems typically consist of the following networked technology in a single room:

- presentation technology (large screen projector, or electronic whiteboard), and
- computation technology (a workstation or portable PC for each participant), and
- group process technology (voting tools, brainstorming tools, etc.)

Notice that in contrast to electronic mail, a GDSS is designed to support real-time face-to-face interaction among people, so it is called same time, same place technology.

Another well-known groupware example is video conferencing, which allows participants in different locations to see and hear each other for a same time, different place collaboration. There are many systems and products of this type available. For example, MBone [40] tools are available and free to allow meeting attendees in different locations to have a distributed meeting over the Internet. Participants in various locations can see (VIC), and hear (VAT) each other, and share a group window on their screens (Whiteboard.) This same time, different place technology is in the same category as MUDs and MOOs (chat rooms,) and group virtual realities.

Our final groupware example is workflow. A workflow management system is a networked control system that assists in analyzing, coordinating, and executing business processes. A workflow management system typically has two sub-systems: (1) A modeling subsystem which allows organizational administrators and analysts to construct a procedural models of the flow of work among people and tasks. This model is embedded in the network system to drive the enactment subsystem. (2) An enactment subsystem which uses the model to coordinate task executions by various participants at various workstations connected to a network. It initiates tasks in their correct order, and keeps track of completed work. Since a workflow system has a representation of work procedures, and knows which actors at which workstations are assigned to do what, it is called "organizationally aware groupware." This is a potentially powerful system which can download appropriate programs and data to users' workstations as needed; assist in task execution; send reminders when and if a user misses a deadline; automatically fill out electronic forms when needed; and generally act as coordinator, historian, and process overseer.

The above groupware examples suggest that there is a wide variety of systems. Some are same time, same place; some are different time, different place like workflow; and some fall in between these. But we see that all of these groupware systems aim to assist people in their communication, coordination, and collaboration.

10.2 Basic Definitions

10.2.1 Groupware

As stated above, groupware is technology to assist groups. Before presenting our elaborated definition of groupware, we discuss the notion of group. In this document, we define groups very generally as collaborating communities of participants. A group may be very small (e.g. two designers working via an electronic whiteboard,) or very large (e.g. all citizens of a large country participating in electronic voting.) A group may be very close knit, sharing goals and tasks and common knowledge and preferences and etc; or it may be a very amorphous group with no knowledge of other group members and no explicit shared goals. This latter type of group is of interest because it is commonly found on the Internet. Terms such as teams, organizations, corporations, communities and societies all fall within our notion of groups; thus groupware may be applicable to these quite varied entities.

When one thinks about typical groupware, electronic mail and video conferencing come to mind as typical examples. In fact, there are many single user tools which have been upgraded to be "group enabled." For example, a single user text editor which has an add-on electronic mail feature integrated into its latest release is groupware, or at least it has a groupware aspect. Thus, when examining the utility of groupware, we must specify which aspects of it we are focusing upon. This chapter suggests a four part classification of groupware according to its aspects.

It is clear that some groupware are much more useful to groups than others. For example, ordinary electronic mail is not as useful as enhanced electronic mail that filters, sorts into various mailboxes, and is multimedia. The filtering helps prevent information overload, the sorting helps to categorize messages into conversations—thereby providing context for messages, and the multimedia allows much more of the group spirit, emphases, and social background to be captured. This document therefore suggests that the question of "Is that technology groupware?" may not have a simple YES or NO answer, but depends upon the aspect of the technology that we are focusing upon, and is best represented by a spectrum. Some technological tools are high on this spectrum, meaning that they incorporate powerful and appropriate aids for group work. Others are considered low on the spectrum because they provide weak or inappropriate aids for group work. Ordinary email is much lower on the spectrum than enhanced email. Fax is also groupware, but it is quite low on the scale. Many group enabled systems tend to be lower on the scale than systems which were initially designed as group support systems.

We are now ready to state our definition of groupware: *Groupware* (also sometimes called *collaboration technology*) is defined as computing and communications technology based systems that assist groups of participants, and help to support a shared environment.

10.2.2 Computer Supported Cooperative Work (CSCW)

CSCW is the name of the research area that studies the use of computing and communications technologies to support group activities. Associated with this are are questions such as "How do people interact and collaborate?" and "How can technology facilitate and enhance this interaction and collaboration?" The emerging new focus on groupware presents opportunities for new paradigms, new types of systems, and new ways of working. Along with these opportunities come new problems and new intellectual challenges. Research methodologies utilized in this area include field studies, laboratory experiments, ethnographic studies, systems prototyping, simulation, and conceptual modeling. There have been a large number of studies, utilizing a wide variety of techniques. The techniques, technologies, and findings in this area have been useful to enhance interactions ranging from real time face to face meetings, to asynchronous organizational workflows.

In this area, it has been the case that technology tends to change and progress at a much faster rate than our understanding of human interaction phenomena. We need a much deeper understanding of the social and organizational factors, and their interaction with technology, than currently exists. There is an important component of the CSCW area concerned with theories, frameworks, and mathematical models. Thus, CSCW includes the theoretical development of models of teams, organizations, and social systems. This effort supports the analysis, prediction, and design of social structures taking into account the participants' information, communication possibilities, objectives, relationships, and incentive mechanisms. In constructing such theories, the area draws upon diverse disciplines including social psychology, organizational design, economics, computer science, and management science. As information technologies drive the underlying factors such as communication possibilities, the theoretical models provide a means to evaluate the effects of alternate designs, and a guide to shaping both the technology and the social systems for beneficial outcomes.

10.3 Aspects of Groupware

In this section we propose a classification of groupware systems that, we believe, is more interesting than previous taxonomies both in terms of its pedagogical advantages and in terms of its ability to direct future research in the area.

Other researchers have proposed taxonomies of groupware systems based upon a same/different time/space distinction [31], based upon areas of application [17], and based upon other criteria such as group size. We propose a classification based upon underlying functionalities of groupware. We introduce four classes, which we call *aspects*. Briefly, the first aspect, *keeper*, groups functionalities that are related to storage and access to share data; the second aspect, *coordinator*, is related to the ordering and synchronization of individual activities that make up the whole process; the third aspect, *communicator*, groups functionalities related to unconstrained and explicit communication among the participants; and finally the forth aspect, *team-agent*, refers to intelligent or semi-intelligent software components that perform specialized functions and help the dynamic of a group.

This classification is neither complete, in the sense that not all functionalities fall within one of such aspects, nor it is categorical, in the sense that it is always possible to say when a functionality falls within one or other aspect. In fact there will be functionalities that seem to lay on the intersection of different aspects. But despite these problems, we believe that the aspect taxonomy is helpful to understand the past and present of the field, and to suggest directions for the future.

It turns out that most current groupware systems have functionalities that fall overwhelmingly within one of the first three aspects. In this sense, we will talk about typical keeper systems, typical coordinator systems and typical communicator systems.

10.3.1 Keepers

Sometimes the collaboration among a group of people is centered on the access and change of a shared set of data. Sometimes the goal of the collaboration is the construction of this shared data, which we will call the *artifact*. The keeper of the artifact, or keeper for short, is the set of functionalities related to the storage and manipulation of the artifact.

Two examples of non computer-mediated keepers are the white board in a brainstorming session in which three engineers are drafting a new circuit, and the draft of a business contract that is circulating among some executives who write their comments about the contract on the margins.

These two examples reflect an important distinction among keepers: there are keepers that allow for more than one user at the same time to alter the artifact, such as the white board, whereas some other keepers do not.

In groupware systems some typical keepers are:

- systems that allow for revision of documents [41]. In such systems, a single person writes a document and then submits it to be reviewed by others. The reviewers may attach comments to segments of the document, or propose changes to it. Then the original author receives the comments, proposals of change, and so on, and changes the document, which may be again submitted to more reviews.
- concurrent editors [16], that allow more than one user to change the same file/document at the same time.
- computer aided design (CAD) and computer aided software engineering (CASE) tools [6].

Functionalities that fall within the keeper are:

- control access rights to the objects. Not all participants have the same rights to the objects that make up the artifact or the same rights to perform certain operations onto these objects. For example in a document reviewing groupware, the reviewers do not have the right to change the real document, they only have the right to attach comments and substitutions to it. In some systems one reviewer do not have the right to read other reviewer's comments, whereas in others, the reviewers can both read and comment each other's contributions.
- control of simultaneous access to the artifact. Some groupware allow for simultaneous changes to the artifact. This poses the problem of maintaining the consistency of the artifact: if two simultaneous and contradictory changes are submitted to the keeper, how will it perform them?
- versioning of the artifact. In some applications it is important to store stable situations of the artifact during the process and to allow the artifact to be restored to such stable situations.
- storage of time stamp and author information on objects of the artifact. Some groupware allow a user to view just the changes performed since she last logged

on, or the changes made by another participant.

- floor control. Some groupware systems use a mechanism of floor control to avoid simultaneous access to the artifact, for example a classroom blackboard. At each time only one user has the right to change the artifact (the participant that has control of the floor). Other users may request the floor which will be granted by the system as soon as the participant that has the floor relinquishes it.

Ontological Model — Model for Keepers

The ontological model is a description of the objects and operations that can be used to construct and manipulate the artifact; the semantics of such objects and how they should be used.

Sometimes, a precise description of how the objects should be used is the essence of a groupware system. That is the case of QuestMap [44]. QuestMap product evolved from the experimental gIBIS [11] system, and as with the other members in the IBIS family it supports decision making by structuring the discussions. It implements the IBIS model of discussion and decision developed during the early 1970 [35]. The IBIS model proposes that decision making about "wicked problems" should be performed in three phases: *divergence*, when solutions to the problems are creatively suggested, *convergence* when after listing all alternatives the group converges to a few of them, and *decision* when all in the group are convinced of the solution to be adopted. Of these three phases the IBIS model considers the first as the most important and proposes that in that phase the participants make a clear distinction between the questions, the solution for those questions, and the arguments in favor or against those proposed solutions. Questions are named *issues* in IBIS and solution are named *positions*.

QuestMap is a tool to support the divergence phase: the discussion is the collective construction of a graphical map that contains nodes to represent issues, positions and arguments; and different links connecting these nodes. By clicking on a node, the user accesses the content of the node: the statement of an issue, position, or argument. Each user can add new objects to the discussion and can delete objects created by herself. All users can access and change the map simultaneously, and their changes will be transmitted to all other participants, or they can access and change the map asynchronously.

Another component of the ontological model is its *concurrency control*: how are simultaneous access and change requests to the artifact dealt with. Some systems would not allow for concurrent change to the artifact because either their collaboration model does not allow for concurrent activities that may access the artifact, or because their mechanism of floor control is restrictive. There are many varieties of floor control. Many systems implement a floor control mechanism in which only one participant at a time has the right to perform changes to the artifact or artifact attributes. Those changes may occur in a fashion that is immediately visible to all group members. A common feature of concurrent systems is the existence of a group window (for example GROVE, a concurrent group editor [16]) in which the same view is displayed to all participants. The region of the artifact that is being viewed in all group windows is controlled by one of many possible floor control mechanisms.

In some other systems, there is no floor mechanism but there is some form of locking: a participant protects a region of the artifact by placing a lock that prevents other users from making changes within that region (for example REDUCE [10]). Some other systems accept all operations from the participants and deal with inconsistent ones in an application dependent way.

Another component of the ontological model is *currency*: how up-to-date are the views that each participant has of the artifact. Depending on the implementation, some systems may present an out-of-date view of the artifact to the participants. In such cases the model may specify a mechanism allowing a participant to request the current version of the artifact, or if this is done automatically, how frequent is the update. Also the model has to specify what happens with changes that are performed on out-of-date views of the artifact. Many options exist. For example, they may remain local, they may be later merged into the current version of the artifact, or they may be sent to the current version immediately. Another issues is how are inconsistencies due to the lack of currency dealt with.

10.3.2 Coordinators

Sometimes collaborating is each participant of the group performing some activity, possibly but not necessarily an individual activity, in a previously defined order. The coordinator of activities, or coordinator for short is the set of functionalities related to this temporal evolution of the system, the enabling of an activity after all its preceding activities are terminated.

A prototypical non-computer-mediated coordinator is the production line in a factory. In the production line, the process of constructing, say a car, was carefully and previously divided into a set of individually performed and temporally ordered activities. This example also shows one of the limits of our aspect model: in a line of production, the point is the construction of the artifact, that is the car. In fact there is almost always some data involved in a coordinator system: people frequently perform their activity upon some data that is passed along to the person that will perform the next activity in line. But we claim that in most coordinators this data is not really shared, the data flows overwhelmingly in one direction, that is, as soon as someone has terminated her activity and performed all the changes in the data, that person will not receive that data again in the process. Keepers to store and control such data are simple (or uninteresting) because the changes to the data are linear and predictable. We claim that for such systems, the coordinator aspect is much more interesting than the keeper aspect, and thus we call them coordinators.

Other non-computer mediated coordinators are techniques for meeting management such as the Delphi method [37].

Some typical groupware with strong coordinator components are:

- workflow management systems [30]
- software process management systems [19, 22].
- some examples of meeting coordinators and group decision support systems [51]

The basic functionalities of a coordinator are centered on the execution (or enactment) of a *plan*, or a sequence of activities (sometimes called a procedure or a process). The coordinator is responsible for insuring that an instance of a process follows its predefined plan. This is also referred as *enacting* the plan or model for that process. Some functionalities related to enactment are:

- enabling an activity once its preceding activities have terminated.
- notification to the users that they may start a particular activity or that a particular activity is late.
- inspecting the current stage of a process. Some systems allow privileged users to obtain various information about the process state, such as which activities have been completed, and when, and by who, and which activities are being carried on.
- dynamic alteration of a process description to cope with surprises. Very few of the existing coordinators allow for changes on the plan of a process. Changes to the plan are important in dealing with unexpected situations, that were not taken into consideration when the plan was conceived.
- helping participants to manage their work. Some systems, such as workflow systems, deal with more than one process at a time. For example, John's purchase order and Bill's travel reimbursement may both be processed under the control of the same workflow system although they are instances of different processes. In such cases there will usually be many activities attributed to a single actor, and the workflow management system may help that actor by displaying the list of activities to be performed by that actor, displaying the deadlines, and allowing the user to choose which activity she wants to perform.

Another important group of functionalities of coordinators centers on defining the plan itself. This is also referred as *modeling*. In general terms, the plan or model is a description of the sequence of activities that should be performed, who will perform them, when they must terminate, and so on. Most coordinators allow for some form of definition of the plan. Meeting support systems sometimes have a predefined sequence of activities, but allow the users to define who will perform them and when should they be finished. Workflow systems and software process management systems allow for the definition of not only who will perform the activities and when, but also what activities will be performed, which supporting tools and environment will be available for each activity, and in what order the activities should be executed.

Coordination Model

The main concept of the coordination model is that of an *activity*. Other important concepts are *role* and *actor*. An activity is a potential set of operations (and the corresponding objects) that an actor playing a particular role can perform, with a defined goal. In general, an actor may be a user, a computer system, or a group. The actor carrying out the activity is called the performer of the activity. A set of activities and the ordering among them make up a *procedure*. Some coordinators are designed for a single procedure, for example software inspection, others, such as workflow systems deal with multiple processes, such as “order processing procedure” and “travel reimbursement procedure” for example.

More than one instance of each procedure may be “executing” at the same time: there may be many order processing jobs being carried out simultaneously in a company for many different customers. Each of this instances of the procedure will be called an *endeavor*.

The coordination model has two components, as indicated above: a component that deals with the modeling of the process and one that deals with the enactment. The plan is a predefined specification on how an endeavor will or should proceed. The plan specifies the activities and their goals, who perform them, the objects and operations available in each activity, the order in which the activities should be performed, when should the activities end, etc. We will call the part of the plan that defines which activities should be performed and in what sequence as the *activity plan*. The component that describes who will perform what activity is the *actor assignment*, and the component that defines deadlines as the *temporal plan*.

The coordination model has to specify which of these components are fixed and which can be set by the user. Some systems have a fixed activity plan, but both the actor assignment and temporal plan can be set by the user. In such systems, the fixed activity plan reflects a methodology that is embedded into the system. Document reviewing systems is an example of a coordination model with fixed activity plan. Other coordinators that have a fixed activity plan are meeting management systems.

Other coordinators allow for all parameters of a plan to be defined by the user. Workflows and software process management systems are examples in this category. Such systems are limited by the language used to define the activity plans, the actor assignment and so on.

The enactment component of the coordination model defines the relationship between the plan and the execution of an endeavor: is the plan a specification of what will happen with the endeavor, or just a suggestion. In other words, is it possible, at enactment time, to change the plan for a particular endeavor or not. This replanning for a particular endeavor may be important to deal with unplanned, unexpected, or exceptional situations. In a workflow system, the order processing endeavor for a particular case may have to follow a different plan than the one predefined because, for example, that customer, which is the most important customer of the company, needs to receive the goods ordered in a very short time.

In this case, the plan for that endeavor may be altered to skip some activities.

Another aspect at the enactment level is whether the system controls/monitors more than one endeavor at the same time. For example a document review system may not be designed to monitor more than one endeavor. In this case, although many documents can be reviewed at the same time, each document is being controlled by a separate instance of the document reviewing system, and each instance does not know about the others. In such a case, the system (or better an instance of the system) cannot know that a particular reviewer is overburdened with five other reviews.

Multiple-endeavor systems may also help users manage their work. Since the system knows about all activities that are assigned to a particular actor, it can provide the actor with information such as which activities are urgent, and which are late.

10.3.3 Communicators

Communication is a basic aspect of any collaborative endeavor. In a mainly keeper application there is (implicit) communication when one participant changes the artifact, and that is known to the others. Also, in a mainly coordinator application there is (implicit) communication when one participant finishes an activity and that enables another participant to start the next activity. But many times there is need for explicit communication among people. The communicator aspect groups the functionalities that allow different users to communicate explicitly among themselves.

Two non-computer mediated examples of communicator are telephone and letters. These two examples also illustrate an interesting distinction among communicators: whether they are same time (real-time) or different time (off-line).

Typical groupware communicators are:

- e-mail.
- desktop conferencing systems (for example [43, 46]). These systems allow a group of people to communicate through audio and/or video from their desktop computers. Some systems allow for all users to both transmit and receive, while others allow only one person to transmit while the others only receive.
- chat and muds/moos (for example [36]). These systems allow for a group of people to interact mainly through text. Participants send their contributions either to the whole group or privately to some subset of the whole group, and each participant sees all messages sent to the group or to her privately.
- white-boards (for example [46, 40]).

Typical functionalities of communicators are:

- sending and receiving a message.
- joining and leaving a conference.

- management help functions and abbreviations, such as mailing lists, alias, and so on.

Conference and Conversational Models

The conference and conversational models are the underlying models of communicators. The conference model describes whether only two or more people can communicate and how that communication is initiated, and if more than two party conferences are allowed, how new people join the conversation, whether it is possible within a multi-party conversation to talk privately to some subset of the group, and so on. The conference model must also specify whether all participants can transmit/receive, and if not how one switches from transmitting to receiving.

The conversational model describes what are the conversational moves allowed in the communication, how participants take turns in performing these conversational moves, what are appropriate conversational replies to the moves, how the groupware can help the user manage each conversation, and manage multiple conversations.

In real time communicators the emphasis is on the conference model. It is assumed that the participants themselves will manage the conversation. For example, in a video-conference system once the participants "get together" in a conference (following the system's particular conference model) it is assumed that the participants will understand when someone's contribution is a question, for example, because the participants will use the group/culture/language appropriate markers and intonations to convey the question. In other words, it is usually left to the participants, and not to the system, to interpret the conversational moves and follow (or not) the appropriate cultural/group protocols for such moves.

In some video-conference systems, such as IVS [29] all participants transmit video but only one transmits sound, while the others listen. Once the participant terminates her contribution she releases the sound control to others. In CU-SeeMe [14] all participants can transmit video and sound, and each participant chooses which video and sound transmissions to receive.

In off-line communicators, the emphasis is on the conversational model. Because there may be a long period between one conversational message and its reply, the groupware system, if it incorporates an appropriate conversational model, may provide help to its users. It may help a user that just received a message to figure out its context, that is, what are the other conversational messages that preceded this one, and what are the appropriate replies to it. The system may help the user by listing all messages that need reply, and what kind of reply is appropriate for each of them, list all conversations that have not yet reached a final state, list all previous conversations, and so on.

Furthermore the conversational model may state that some types of messages need no reply but should be processed automatically. For example, let us assume that the conversation model specifies that a message of acknowledgment of receipt should be sent in response for a message in which the field "acknowledge-receipt" is set. The communicator not only can send this acknowledgment upon receiving an incoming message with the field set, but can also process incoming acknowledgments and alert the user of messages that the user sent more that 2 days ago and for which there has been no acknowledgment.

An communication system that has an elaborate and explicit conversation model is The Coordinator [52, 21]. The Coordinator implements Winograd and Flores' model of conversation for actions. In the model a conversation is started by a request (to do something, before some time). The recipient may accept the request, may refuse it, or may negotiate. If the request is accepted, maybe after negotiations, and performed, the recipient of the request declares it completed, which is accepted or not by the original sender. Each message identifies itself as a particular message type called a "conversational move" in an ongoing conversation. For example the user would understand that a particular message is user B's modified request after a first round of negotiations and that this user appropriate response would be to accept, reject, or re-negotiate this new request. Furthermore, The Coordinator would assist the user to manage her obligations: the user would be able to list which requests from others she has accepted and still has to perform, and when are they due, which request the user has not answered yet, which of her requests has not been performed until now and so on.

The clarity of the Coordinator's conversational model has caused a large impact in groupware research, and there has been much discussion, debate and study about the usage of this type of system [49, 15].

10.3.4 Team-Agents

Team agents are artificial participants that perform specialized functions within a group setting. Besides groupware modules which must be concerned with the operation of the entire groupware system, there are frequently modules which are built to perform specific non-global subtasks. These frequently involve specialized domain knowledge; we call these modules team agents. Examples include the "performance specialist" within a software engineering team, and the "social mediator" within an electronic meeting. Neither of these examples is concerned with the overall workings of the system, but each contributes useful functionality in a specialized domain as part of a group. Thus each is a team agent. Ideally, team agents act as if they were full fledged, actively participating members of the group.

An important distinction within the category of team agents is *autonomous agents* versus *single user agents* versus *group agents*. Autonomous agents primarily work alone on an independent subtask; single user agents (e.g. user interface agents) interact with, and work for a single participant within the group; group agents interact and collaborate with the various members of the group as a true colleague. Group agents thus need a good understanding of the goals, structures, and personalities of the group, and of their role within the group.

Group Critic

Some (single-ware) computer aided design (CAD) systems have critics that comment or check the user's designs. Critics are AI programs that tap into an artifact being developed and reports problems with the design. For example, the critic described in [20] warns the designer of problems in kitchen design such as a stove too close to a window and so on. Although at the time of the writing of this chapter there was no group-CAD that incorporated critics, it is conceivable that they will in the short future. Such critics would be good examples of team-agents.

As a team agent, and more specifically as a group agent, a group critic must be aware that the problems it find in the design are the result of different users acting on different goals and all are responsible for the problem. For example, if the critic detects that the stove is too close to the window it must warn the user that placed the window and the user that placed the stove, even if placing the stove was done last.

Appointment Scheduler

A popular groupware application is group calendaring and scheduling of meetings [50]. Such softwares allow one to schedule a meeting among a group of people by selecting a free time slot for all meeting participants. In order to do that, the scheduler must have access to each participant's individual calendar. An interesting scheduler would also know about peoples' preferences for meeting hours, and in case of a cancellation of a meeting this system could re-arrange some of the meetings so the participants would be happier with their times.

The appointment scheduler, specially the implemented ones are mainly autonomous agents. But depending on the functionalities it may also be a single user agent. An appointment scheduler that knows about its user's preferences and pro-actively tries to satisfy those preferences is certainly acting on behalf of its user.

10.3.5 Agent Models

It is important to notice that the use of the term agent in this chapter is broader than its use in most of the other chapters in this book. Agent is any automatic process; it does not need to be "intelligent" or "autonomous" in the sense used in other chapters.

In particular, for the purposes of this chapter, an autonomous agent is a program that runs independently, and has no interaction with any user. An autonomous agent may tally the votes in a decision meeting, it may compile a program in a software development workflow, it may print an acceptance letter based on a template and data available in a database, and so on. None of these activities are considered intelligent. But an autonomous agent may choose a particular methodology and tool for a meeting, based on the problem [1], or another agent may plan the sequence of activities to be performed based on the goals to be achieved

[12]; these are more “intelligent” activities.

A model of an autonomous agent will not be developed in this chapter. The theories and models put forth in the other chapters are all relevant to define, classify and model autonomous agents.

Group agents are programs that interacts with all participants and thus “should behave like a participant” therefore group agents should incorporate a model that at least describes what is “to behave like a participant.” But there are no implemented group agent and thus there is not enough experience to abstract into a group-agent model.

User Agent Models

Groupware reflects a change in emphasis from using the computer to solve problems to using the computer to facilitate human interaction. Users can best take advantage of this changed emphasis via systems with user-interfaces especially designed for groupware. We call these group user-interfaces. The issues that designers of group user-interfaces face are challenging and are significant extensions of the usual issues of interfaces for single-user systems. Thus the user interface conceptual model is highly concerned with representation of human-human interaction, and significantly transcends single user interface models. The model has four components:

- views of information objects and operators
- views of process and communication;
- views of participants;
- views of shared context.

Firstly, the user interface for a participant in a groupware session must be capable of presentation of the objects and operations embodied in the ontological model as previously defined in this chapter. Since different participants may have different abilities (or different perspectives), the user interface model includes the concept of multiple views of objects and the concept of local operations which are typically not present in single user models.

Besides the ontological model data, the user interface in a groupware system may have to deal with other “meta objects”. Examples include telepointers and group windows [16]. For example, on systems that allow for inspection of the stage, the user-interface has to display this information.

Views of objects derives from the fact that different participants may want to have different views of the same objects of the system. For example, in a GDSS, an object may be semantically an array of numbers, but one participant may opt to view it as a bar chart, another may prefer to see it as a pie graph, or as a table. All these different representations are views of the same object, and in principle it should be an issue related to the user interface. Furthermore, if one of the operations allowed upon this generic array object is to alter the values of its elements, the operation has to be translated appropriately to each view of the object. In the bar chart view of the table, one could change the corresponding element by stretching the height of the bar corresponding to the element. In a table view, one could type the element index and change its value using the keyboard.

Similarly, in an IBIS one participant may want to view the network of issues, positions and arguments as a graphic network of connected nodes. Another may want to view it as linearized text, indented appropriately to differentiate issues from positions from arguments.

The concept of local operations derives from the observation that group editing is a common and necessary operation in many groupware systems, but all edits need not be seen immediately by all participants. There are decisions or options that must be built into each groupware system concerning granularity of edits and locality and when to transmit to others. If an edit operation is part of a real time synchronous interaction, then a WYSIWIS ("what-you-see-is-what-I-see") system may transmit the edit immediately to all participants. Alternatively, within an asynchronous system, edits may not be transmitted to other participants until a save operation is executed. For many existing systems, when entering data through the keyboard, the system considers all key presses as local operations until the return key is pressed. Thus composing a line is an atomic operation. Finally, it should be pointed out that in some systems, operations on meta objects, e.g. pointer movement, are permanently local operations which are never transmitted to other participants.

In a group editor system, for example, if the participant realizes that he mistyped the last character, he would press the backspace key and erase it. If we were to consider that within a real time interaction, it might be transmitted immediately. In GROVE and the Unix talk program, backspace key presses are object level operations, and are transmitted immediately. Clearly the level of granularity of operations depends upon the application and the group environment.

In a synchronous system, understanding who is simultaneously doing what is useful, and should be presented to users. In an asynchronous system, it is useful to understand who did what since the user last signed on. This leads to the notion of "view of the process." Workflow systems are a clear example where it is useful to have answers to questions of what step preceded mine, and what follows. The user should be able to see upon demand, a simple view of which workcases are in which stages. As exceptions arise in processing, formal and informal communication ought to be facilitated. Maps of who holds which positions, and who talks to whom help to make communication visible.

Groupware is much more concerned with assisting people to people communication than single-user systems. Providing some convenient means of knowing other participants, and what they are doing is an important aspect of our model. The identity of a participant is not directly related to the completion of the endeavor, but this information can be extremely helpful to the other participants in evaluating the situation of the group dynamics. For example, knowing that Smith is in a group long-distance discussion mediated by an IBIS, may lead the other participants to formulate their contributions in different ways than if Smith was absent.

GROVE, for instance, provides this context information by displaying the pictures of the participants at the bottom of each group window. Systems of video windows, video walls, virtual rooms, and virtual realities may display the real time video images of participants which helps with the evaluation of everybody's attention and mood during the session. ClearBoard, for example, allows shared video drawing, while super-imposing the image of the collaborating colleague [28]. This allows eye contact and gaze awareness, while still focusing on the work artifact.

In addition to displaying participants, it is possible to present, in an unobtrusive manner, relevant status, background, and preferences of participants. Benford discusses concepts of auras, nimbus, focus, and adapters [3], all of which are within the scope of the group user interface model. Group information such as the social network of who talks to whom can be presented, and the view of this can be tailored to the viewing participant. For example, if there is a relevant and significant shared previous experience between Smith and me, then I would like to be reminded, and associate this with Smith.

Other possible forms of participant context information are: group opinion on relevant issues, extent to which participants know each other, status of the communications technology, how are the participants geographically distributed; information from a database on relevant aspects of each participant; etc. The information on the geographical distribution of participants may help long distance participants to realize that the subgroups that are in the same place may have developed other protocols of communication besides the one enforced by the collaboration model of the system. Remote response time information may help the division of labor among the group members so that the tasks of a long-distance participant should not depend on high currency. Finally, information about other participants may help a user to place the context of the other participants contributions.

Another area of presentation that should be dealt with by a group user interface is all of the useful background material that we call context. The choice of what and how to present contextual information is a challenge, and that context may include items as diverse as the time of the next meeting, the current weather, and the presence of new mail messages from other participants.

We categorize contextual information as structural, social, or organizational, Structural context includes what and where data, such as the set of interactions in which I am currently participating, and temporal information such as what data has changed since I last accessed this hypertext web. Within a software engineering project, useful context may include languages and case tools used, status of various code and documentation files, and future milestones.

Social context includes items such as group norms, group metrics, and social history of the group. One proposed metaphor of shared virtual reality is that different projects would take place in dramatically different virtual rooms. Thus, the group's difficult design project would take place in a Tahitian hut, and the formal election processes always takes place in a London House of Lords; just the act of re-entering these contexts might trigger much useful contextual information in the heads of the participants. The research work on GroupAnalyzer [38] explored the efficacy of providing an electronic meeting barometer for groups in face to face interaction. This is an excellent example of the utility of graphical context presentation.

Organizational context can apply to small groups, large corporations, countries, or international organizations. It potentially includes formal reporting and responsibility structures of the group such as the organization chart. Also included are other items such as rules of the organization (procedures manuals, etc.,) and inter-organizational data (competitive edge, mergers, etc.) In general, it must be understood that a meeting or any interaction is not an event in isolation, so these contextual clues provided by the user-interface can make the difference between a successful interaction versus a failure.

Finally, we note that participants are not context! Context connotes objects and conditions that are in the background. A primary function of groupware is support of communication and collaboration among participants, so participants are in the foreground.

10.3.6 An Example of Aspect Analysis of a Groupware

Let us discuss some different possible implementations of whiteboards which mix aspects from communicators and keepers. Whiteboards are group drawing tools, somewhat like a group Paint. Whiteboards have a strong communicator component, specially the conference model. People may join and leave a ongoing whiteboard session.

The simpler whiteboard has one cursor for each participant. The participant by moving the cursor around sets some pixels to, say, black, and each participant sees the composition of all contributions. The canvas can be seen as a keeper, but a trivial one. The objects maintained buy the keeper are pixels and the only operation available is to set them on. There is no problem of concurrent access to the same pixel, and the currency of each participant's view is not critical.

In a more elaborate whiteboard, each participant still has her own cursor but each cursor paints the canvas in a different color. The objects maintained by the keeper are a little more complex, pixels with colors, but still very simple. An even more elaborate whiteboard in terms of its keeper model would be one in which each participant paints on her own transparent canvas, and each participant can choose whose canvas or canvases she wants to see. Now the ontology model is yet more complex. Finally, let us assume that the system is also used asynchronously, and that each participant can choose to see only the changes made by some particular participants since she last logged into the system. Now the ontology model includes objects like canvas, time and author stamps, and so on. But all these versions of a whiteboard are simple in regard to object coordination, or concurrency control.

A whiteboard system that is more complex in terms of concurrency control, even though it has a very simple ontological model, is the first whiteboard modified so that there is only one cursor for the whole group. The cursor is the resource that needs to be controlled by the keeper, and some way of passing this control must be planned. It could be a first come first serve floor control with explicit release (the one that holds the cursor must explicitly release it, and control will pass to the first one waiting in line for it), or release by timeout (after a period of inactivity the cursor goes to the next in line). Or the cursor may be owned by some privileged participant that may pass temporarily the control of the cursor to one participant, but may regain its control any time she wants.

10.4 Multi-Aspect Groupware

The aspects model of groupware is interesting because not only does it serve as a guide to the designers and users of groupware systems but it also allows for a perspective on the past research on groupware and we believe can point the way to future research in the field.

Most Groupware research done until the 90's were single-aspect systems, that is, a system in which functionalities within one aspect overwhelm the functionalities within other aspects. But there are some exceptions, such as document reviewing systems (for example [42]) which mix keepers with coordinators. Another such system is The Coordinator [52, 21], which mixes communicators with coordinators.

We will describe below the Chautauqua workflow system, which mixes all the four aspects described above.

10.4.1 Chautauqua — A Multi-Aspect System

Chautauqua is an Internet based collaboration management system designed and implemented within the Collaboration Technology Research Group [13] at the University of Colorado, USA, and the Center for Informatics (ZID) at the University of Arts, Austria. This exploratory prototype, which has been in test usage since 1995, illustrates the possibilities and advantages of tight integration of coordinator, keeper, communicator, and agents. At its base, this system is a workflow management system. However, unlike conventional workflow systems, this system carefully incorporates functionality for goal based reasoning, for real time interaction, and for flexible, human controlled dynamic change [18].

The history of workflow products in corporate America has been mixed; more systems have silently died than been successful [24]. Workflow has been heavily criticized because of its typically inflexible and dictatorial nature compared to the way that office workers really accomplish tasks. Chautauqua attempts to address these criticisms by being strictly a subservient system—it incorporates novels features including flexible exception handling mechanisms, representation of inconsistent concurrently updated information, assistance for simultaneous group editing, and powerful, verifiable dynamic change capability. All of these features are accessible to any and all users with appropriate access rights.

Thus, the information concerning procedural specifications associated with the coordination aspect, which we call organizational awareness, is the artifact maintained by the sophisticated Chautauqua keeper. This information is available (in graphical, easy to use form) to all users for seeing and understanding the procedures, and also for making changes to the procedures. The keeper must support simultaneous editing of this information, and must be capable of mediating and merging inconsistent information entered by different users. The keeper is thus integrally integrated with the coordinator.

Techniques implemented by Chautauqua for assisting with the above mediation include the concept of "town meetings" and "group decision sessions". Clearly some of the tasks such as problem solving and decision making can be facilitated if the system can schedule and initiate real time video conferences. Being organizationally aware, the workflow system is in a good position to do this. Thus, Chautauqua integrally integrates a communicator.

The dynamic change feature goes far beyond application data update capability to allow open change to control flow and to organizational structures. Furthermore this change can take place in the midst of system enactment without stopping and restarting, or aborting work cases in progress. This feature is implemented via "Change Agents" that have certain global knowledge about the state of the executing system and which work cases are where. Consider a procedure in which task A is specified to execute before task B. If the dynamic structural change is to respecify that A and B should be done in parallel, then work that is "inside of task A" at the time of change (the change taking effect immediately and instantaneously) may accidentally never execute task B. This is a simple example of potentially complex inconsistencies that can occur if dynamic change is not carefully managed.

In general, the change agent can work with the users and utilize its global knowledge to analyze changes for potential problems. Note that a change can be permanent, or a temporary one time change. Thus, exception handling falls within the dynamic change category. The change agent in Chautauqua is integrally woven into the design of the Chautauqua system, and uses an analytic method based upon graph grammar rules applied to Petri nets to do this change analysis efficiently and effectively. This is yet another example of the importance of designing in the multiple aspects of a groupware system rather than attempting to add on an agent or a keeper as an after-thought.

10.5 Social and Group Issues in Designing Groupware Systems

On designing a groupware system, one has to be aware of multiple levels of issues. At the top most level, one has to be aware that because groupware systems deals with groups, intuitions and experiences appropriate for singe-ware may not be appropriate for groupware.

For example, [23] discusses that group appointment schedulers are used in real work situations far less than the intuitions of someone that had to schedule a meeting among a couple of people would suggest. Grudin suggests that the explanation for that is that in real, hierarchic work situations there is a strong separation between the people that benefit from the existence of the system (the ones that can call a meeting) and the people that, because of the system, has to do extra work (the ones that have to enter their schedules into the system). If someone fails to mark all her appointments in the system she risks the possibility that the system will schedule a meeting during those unmarked but otherwise busy time slots. On the other hand, it is very to sabotage the system by blocking all time slots; by marking all time slots as busy, one will not have to enter one's detailed schedule, nor risk having a meeting scheduled at a busy time slot.

Thus for such a system to be successful the users have to have an incentive to use the scheduler by itself, despite its group benefits. Thus the real issue on the success of failure of a group scheduler is not centered in the group part: finding a free time slot, communication protocols that would allow users in different computer to schedule a meeting and so on. The central issue is providing functionality that pleases the user when using the system as a single-ware. If that is resolved, then the designer has to deal with the next level of issues, in this case the ones that relates to the group functionalities. If the user decides to use the system as his calendar tool, then this user will probably not be happy if his daily appointments were made known to other users when they try to schedule a meeting with her. Thus the issue of privacy, which is a very important issue when dealing with groups, becomes relevant at this level.

Because groupware systems have to be used by all participants, there is an all/nothing or sometimes critical-mass characteristic to the adoption of such systems. Using the scheduler example above, if one of the team members decides not to use the scheduler, either because she does not like the user-interface or the functionalities, the whole team cannot use system as intended.

In other cases the issue is not whether a whole team adopts or not a groupware system, but whether a single user decides to adopt a new technology that would allow her to participate in some collaboration. This shows a critical mass characteristics: it is only attractive to adopt a new communication technology, say voice electronic mail, if enough of the people one wants to talk has voice electronic mail.

In order for a groupware systems to be adopted and accepted its designer must be aware of the issues above, but even more important, he must be aware of how the people for whom the system is being build really work. There is a difficult and moving line between wanting to improve how people work together by means of a groupware system and not violating how the work is done without such system [15].

More and more research reported in Groupware conferences are analysis of work practices, the influence of technology in these practices, the influence of a particular groupware system on a team, aspects of the adoption (or not) of a groupware system, and so on. For example [5] describes the use of anthropology to understand the work practices of a group as part of the methodology to design a particular groupware system. [47] describes the two distinct views about work in organizations and its impacts on the design of groupware systems.

10.6 Supporting Technologies and Theories

We believe that the espoused taxonomy of keepers, coordinators, communicators, and agents is not only useful at the application level, but also at the middleware (resource managers, protocols, etc.) level and at the underware (hardware, basic resource providers, etc.) level. In this section, we discuss supporting technologies, where technology is interpreted in the broad sense to include hardware underpinnings, software underpinnings, and conceptual underpinnings. Thus, communications hardware, software construction kits, toolboxes, protocols, and underlying theories are included.

We also note in passing that any lower level category of technology may be useful for the implementation of several different categories at higher levels. Thus, for example, low level technology such as Ethernet, which is in the communicator category, is very useful and important to implement higher level communicators, but also to implement items in the high level coordinator category. Otherwise there is no vehicle for synchronization signals to get from one module to another.

10.6.1 Keepers

At the bottom underware level there are numerous examples such as RAID disks and CD-ROM technologies that form data storage underpinnings for groupware applications. At the middleware level, examples include file and database systems, particularly distributed ones. We should also mention conceptual middleware such as object oriented and relational database schema technology. All of these technologies help to support generic application level groupware such as organizational memory and electronic librarians; also groupware in the keeper category targeted toward a specific application, such as group CAD systems.

10.6.2 Coordinators

Coordinators may range from workflow systems to GDSS to the UNIX Make software. Middleware that greatly facilitates the construction of this includes the ISIS synchronizer [7], workflow meta-systems such as ADONIS [33], and at a lower level, network operating systems. At the lowest levels, we find interrupt hardware as a primitive that handles coordination at the lowest level. Kernel schedulers are software just above the interrupt hardware that would also be categorized as coordination underware. Although ATM transmission technology is within the communicator category, the ATM switch is a sophisticated technology that is strictly concerned with synchronization—it is within the coordination underware category.

Some of the conceptual underpinnings for coordination seem to fall close to the boundary between middleware and underware. We feel that the speech act primitives [48] are indeed conceptual primitives and fall within the underware category. On the other hand, process description languages such as ICNs, and fundamental models of coordination such as Petri nets seem to fall within the middleware category because they are really concerned with the description and management of coordination.

10.6.3 Communicators

Communicators at the application level may range from generic email and video conferencing systems to very application specific systems. Increasingly more and more remote collaboration is performed over the Internet using the MBone technology [34, 40], and modern successors to it [46]. On the one hand, the specific tools for MBone video, audio, and whiteboard are at the application level. On the other hand, the underlying multicast protocols are middleware, and the hardware systems which allow implementation of efficient MBone multicast protocol are underware. At the communication underware level, many examples such as Ethernet exist. And below this there is much work on wireless transmission, and on satellite transmission, and etc. As previously mentioned we definitely consider conceptual technologies within our categories. Thus, there are many communication protocols which fall within the communication middleware category. The well-known ISO seven layer communication protocol is an example. Of course the lowest layer of this is clearly underware, and the highest, 7th layer, is clearly and strictly at the application level, but all other layers are middleware. Finally, we remark that the Internet and the WWW when viewed from our taxonomy, are not synonymous with groupware (contrary to some vendors claims.) In fact the Internet is simply one of many possible communication vehicles, and the various parts of this technology need to be placed in various sub-categories. Thus, HTTP, the hypertext transport protocol, is simply one of many choices of middleware for implementation of groupware. This needs to be clearly distinguished from HTML, the hypertext markup language, which is not concerned with communication but with presentation to the user. It thus falls in the category of user agent technology which is discussed next.

10.6.4 Team-Agents

As previously described, we divide agents into categories of autonomous agents, (single) user agents, and group agents. The other chapters in this book describe both technological and conceptual underpinning of the agents. In the domain of agents supporting users, UIMS's (user interface management systems) and user interface implementation toolkits have been well used to construct sophisticated user agents.

An interesting category which is much less visible is the group agent category. However, one domain where rapid progress is being made is virtual reality for groups of participants. These systems allow multiple participants connected via a network to a virtual reality system to also see (or sense) and interact with each other. These systems sometimes implement the metaphor of a room, a shared desktop, or perhaps a castle or dungeon with dragons. New middleware that can be used to implement these types of systems includes NetEffect, a distributed server based toolkit for multi-user virtual worlds on the Internet [8]. As conceptual middleware, HTML and VRML (virtual reality markup language) are available. At the underware level, multimedia hardware and virtual reality hardware are proliferating. We see this team agents category as an under-represented one where exciting research and development is now happening.

10.7 Other Taxonomies of Groupware

10.7.1 Space/Time Matrix

[32] classifies groupware systems based on the same space, different space, same time, different time distinction. GROVE and IVS for example, would be both same-time, different-space groupware. Sometimes this distinction is more profitable applied to certain activities or functions within particular groupware rather than to the system as a whole. Let us consider a software inspection system which supports a single programmer writing the code, supports the simultaneous and concurrent inspection of the code by three reviewers that attach comments to segments of the code, and supports the programmer changing the code to suit the reviewers' comments, or discussing with that reviewer why the code is correct as it is, and finally supports the reviewers verification that all their concerns were addressed by the programmer. Is this system same-time or different-time? In fact, its both! But at different stages. This is becoming more prevalent as the needs of groups are becoming better understood.

Furthermore the distinction of whether the activities were performed at the same time or at different times is sometimes not appropriate. In the example above, one could assume that the system would also support that the review activities were performed at different times. What is important is whether the system necessarily requires some activities to be performed at different times, or necessarily require that they are performed at the same time. In the example above, the activity of reviewing and the activity of changing the code had to be performed at different times. That may be because limitations on the keeper component of the application, or it may be because the coordination model so requires: it is reasonable to require that the programmer should not have access to the reviews until they are done, in order to avoid unnecessary argumentation and confusion.

10.7.2 Application Area

It is also reasonable to classify groupware systems according to application domains [17]: group editing and reviewing, workflow, group decision support, real-time communication, distance learning, etc. It turns out that groupware systems for some of these application areas fall clearly within one or another aspect. Group editing and reviewing, for example, are typical keeper systems. Workflow systems are typically coordinators.

The application area in which there is some interesting diversity in terms of aspects and models is group decision support, or meeting support. In this area the goal is to provide methodological and technological support for meetings.

In some way meetings can be seen as the most unstructured form of collaboration; however activities within meetings, such as voting, are frequently very rigid and structured. Some systems propose to support meeting by temporally structuring well defined activities, and thus are mainly coordinators with fixed activity plans. Each activity has some predefined goals and need some specific tools, which may be communicators or keepers. For example, many meeting methodologies propose at least an activity of brainstorming [25], in which all participants are encourage to contribute many possible solutions to the problem at hand. An appropriate tool for such activity is a backboard-like communicator, to which the participants submit contributions, usually anonymously [43]. A few other systems, such as those in the IBIS tradition, choose to structure the meeting "spatially" and thus are mainly keepers.

10.8 Groupware and Internet

At the time of the writing of this chapter, the word Groupware is frequently accompanied with the word Internet especially in the non academic press. This section tries to elicit the relationship between these two concepts.

In order to do that, we need to model the architecture of a typical groupware system/application. The user will interface with the system through a software component that we will call *user software component* (*USC*). The USC runs in the user's computer, but its functionalities are varied: the USC may include all or just part of the user interface, it may host part, or the whole, or a version of that artifact if the system is mainly a keeper, and so on. The USC must communicate/exchange information with other software components that are, typically, in other computers. Thus if the groupware is a teleconference system, each USC needs to communicate with other USCs and there may be a server component.

In order for the USC to communicate with components in other computers it must make use of services provided by another software components, the *network software component* (*NSC*). It is unimportant whether the NSC is part of the operating system of the user's computer, or if it is implemented explicit as part of the USC. The relevant issue is that the NSC runs in the user machine and it is either provided by that machine or implemented in the USC itself.

Given this abstract architecture of a groupware system, one can elucidate two possible relations between Groupware and the Internet. The first relation, which we call *Internet as infrastructure*, the Internet implements the NSC. In the second relation, which we call *Internet as presumed user software*, the Internet implements the USC.

10.8.1 Internet as Infrastructure

The idea behind the Internet as infrastructure is that computers that are on the Internet must have software that allows for some/all of the functionalities required from the NSC. A computer is "connected" to the Internet when it is able to send and receive information according to the many standard protocols (UDP, TCP, FTP, SMTP, NNTP, HTTP, and so on).

Thus, by using the user's computer's own Internet software as NSC, the designer of the groupware avoids having to design the groupware's own NSC (which may be extremely difficult since the services provided by the NSC must necessarily interact with the computer's low level software and hardware), or avoids having to assume that the user's computer has a particular non-standard NSC incorporated into the operating system.

First, it is important to notice that using the Internet as NSC is not the only option available for the designer: computers in a local network may use other protocols that are not available in the Internet; even if the computers are not in a local network, the designed has the option of having one computer phone the other and use non-standard protocols over this phone connection, or the distant computers may be part of a private non-Internet network with its own protocols. But in all these alternatives, the designer has to assume that the user computer will be on the local network, or will have access to a phone line, or will be part of the private wide-area network, and in many of the cases the designer will have to write the NSC herself.

The Internet has some peculiarities that must be taken into account by the groupware designer: it is unreliable, that is messages may not get to their recipients, and it is insecure. Furthermore, at the time of the writing of this chapter, the Internet is still weak concerning protocols for real-time transmissions, and multicasting.

10.8.2 Internet as Presumed Software

The second frequent relation between Internet and groupware is that the Internet implements the USC altogether. One Internet software popular for this task is the WWW browser; at the time of the writing of this chapter, to "be in the Internet" is to have access to a WWW browser. By implementing the USC by a WWW browser, all Internet members are potential participants in the groupware. By pointing the browser toward the server in which the applications runs, one becomes a participant in that system.

The advantages of implementing the USC by a WWW browsers are many for the user: there is no need to buy/install a separate USC to become a participant in a system; the user has to deal with a single interface, the same software (WWW browser) can be used for many applications.

But using a WWW browser as the USC has some problems and limitations. The limitations can be separated into three categories: user interface, client-server architecture, and system interface limitations [4].

The WWW browser can display a large variety of information in different media: text, different picture formats and so on. Furthermore for most browsers it is possible to define external programs that will display media types that are not internally dealt with by the browser. These media types include many audio and video formats and also application specific formats, such as postscript, and text editors internal formats. But as an input device, the WWW browser is very limited; it accepts text typed into the fill-in fields, selection of radio buttons and predefined lists, and mouse click on predefined regions of text and on predefined images. No other gestures, such as mouse movement, and single and double clicks are supported. This poses some severe limitations for some applications; it is not possible to write an application in which the users freely draw a diagram, or changes a diagram by dragging some of its components, or build a diagram by placing predefined shapes onto a canvas.

The client-server limitation is also central for the designer of groupware systems. WWW was designed as a typical client-server protocol: it is stateless, that is the server does not remember any of the previous history of communication with the client, and only the client can initiate communication. The first limitation has been long solved in the WWW community either by using cookies [45] or by encoding the state of the communication in the information sent by the client to the server. When the client requests new information, the encoded state is also transmitted and used by the server.

The real problem is the fact that the WWW server was originally designed as "pull" technology. This means that the server cannot initiate communication with the client. This limits applications in which the currency must be high. In a keeper application, for example, if other participants change the artifact, the server cannot warn the client that the artifact has changed unless the server is "push" technology.

The user system limitations are less severe. They refer to the browsers ability to communicate and operate changes into the user's system. WWW browsers typically allow for files to be downloaded, from server to browsers, but not uploaded.

If the user has locally made changes onto the artifact, using software that runs on the user environment, it may not be possible to easily upload it to a central repository or to the other users

The use of JAVA seems to be a solution for all this problems, but at a price. By writing a JAVA program (or applet) that runs on the user's environment the groupware designer is able to overcome the three problems mentioned above. For example the designer may write a program that accepts user's gestures as inputs, or a program that communicates with other components of the system using its own protocol, instead of using the client-server model underlying HTTP. But in the first case the designer will have to write at least the interface components of the USC, and in the latter case, the NSC. The advantage is that the designer does not need to make presuppositions about the user's environment; if JAVA indeed becomes a widely accepted standard then the designer may assume that the user's WWW browser will both be able to run the program and serve as interface to the user's environment. Thus the same USC program will run in different hardware, operating systems and network environments.

10.9 Conclusions

The authors believe that future research on groupware will be centered on agents and multi-aspect groupware. There are many forms of mixing aspects into a single system. Let us discuss some.

10.9.1 Incorporating Communicators into Keepers

Sometimes participants working in a keeper must communicate to each other directly to understand points of view, to synchronize actions and so on. Some of these communications will be about the artifact and for these communications it would be important to be able to refer to parts of the artifact within the communication, in similar ways that pronouns such as *this*, and *that* are used in conversation to refer to outside entities.

This could be accomplished, for instance, by incorporating a video conference tool within a keeper that would allow synchronous communication among participants, plus some form of telepointer, which would allow the participants to make references to parts of the artifact. The solution is less clear if one also wants to provide off-line communication among the designers.

10.9.2 Incorporating Keepers and Communicators into Coordinators

There is a very common view associated with coordinators that the whole process should be decomposed into sub-processes and so on, until one reaches an "atomic" sub-process, usually called a task or an activity, which cannot (or need not) be further decomposed and which is to be performed by a single actor. The model of the production line is a good example of this "decomposition until the individual task" idea. But there are activities which should not be further decomposed and are not individual. In a workflow system, one may have, for example, a "write proposal" activity which is to be performed by a team. This activity should not be further decomposed into smaller activities because that would constrain too much the team creativity to write the proposal.

To support these collective activities within a coordinator, one needs an appropriate keeper. For example in the "write proposal" activity one needs a keeper that would allow all the actors to work on the proposal. Probably one would also need communicators so that the participants can interact beyond the limits of the ontological model of the keeper for that activity.

10.9.3 Future Research on Agents

Future research in agents in groupware will follow two directions: (autonomous or group) agents to act on the domain and agents that act on the interaction. Domain agents are similar to the group critic described above: they know about the domain of the collaboration. The kitchen group critic knows about kitchens and stoves and so on. A group critic on bridge building may know about material strength and stress; a group critic on software development may know about coding and naming conventions, or about proving a program correct in relation to its specification, and so on.

An interaction agent does not need to know about the domain of the collaboration but knows about interaction/collaboration itself. These agents could play a role similar to human facilitators in meetings. Such facilitators do not necessarily know about the topic that will be discussed in the meeting but they understand meetings, they know when the discussion is becoming too polarized, when an intermission is appropriate, when there are people and views that could not be expressed because of the dynamics of the meeting, and so on.

An interaction agent would analyze the state of the interaction and propose activities for the participants. For example, given the number of alternatives to answer an issue, an interaction agent may propose one of many voting procedures. Or given the statistics of the messages being exchanged (mainly from a few participants), the agent may propose a different methodology for the discussion. More ellaborate interaction agents may understand some aspects of the messages being exchanged and help to categorize the group in terms of its group dynamics [2].

10.9.4 Future Research on Keepers

Another area in which future development would be interesting are the specifiable keepers. Keepers embed an ontology model that is usually fixed, and defined a priori by the groupware designer. But it would be interesting if the users themselves could define or adapt the ontology of the keeper. In the same way that workflow systems are specifiable coordinators, there is a need for specifiable keepers.

In fact Lotus Notes [39], is a specifiable keeper. In a simplistic view, Lotus Notes is a free form database; users (or the system administrators) can define and provide semantics (attach programs) for particular fields of the documents and thus adapt Lotus Notes to particular applications. However, it is not at all easy for naive users to define and adopt.

There is very little research on the languages and primitives appropriate to specify or parameterize an ontology model.

10.10 Exercises

1. *[Level 2]* CSCW and Groupware typically complement and help each other. However, sometimes CSCW studies technological mismatch in which certain groupware is inappropriate for certain work situations. Concoct a work situation, and a groupware technology in which this mismatch might occur. Explain why you think that the groupware technology is inappropriate for your concocted work situation.
2. *[Level 2]* Define the terms collaboration, cooperation, and coordination in a way which you feel is clear and useful for the CSCW community. Justify your definitions, and compare / contrast the three terms.
3. *[Level 2]* This chapter also introduced a time-space taxonomy. Criticize this taxonomy. Should there be any further cells than the four which are presented? Are there any further dimensions which should be added to the simple 2X2 matrix?
4. *[Level 2]* Two family of applications were not mentioned in the aspect section. They are multi-user action games, such as DOOM and QUAKE [26], and shared window [53]. Discuss if these systems fall overwhelmingly into one aspect or another, of if they are multi-aspect systems, of if they show that the aspect classification is not complete because it does not capture what is essential in these two softwares.
5. *[Level 2]* There are a few WWW-based chat services available. In most of them, the user fills a form with her utterance and after submitting it, she receives back a page with all recent interactions in the chat. Thus a user cannot "lurk" at the chat, that is, listen to the other conversations without making herself a contribution. But some chat pages allow for real-time chat, in the sense that the page will change to reflect the recent interactions even though the user has not send her contribution. Which non standard WWW technology does these pages use, and what is the network impact of using them.
6. *[Level 2]* Besides WWW-based chats there are many other examples of Groupware applications that use WWW as infrastructure. In particular INOTE [27] is an image annotation system, and BSCS [9] is a central repository of artifacts. Discuss which of the problems mentioned in section 10.8.2 these systems faced and how did they solve them.
7. *[Level 2]* There is a view that the collection of WWW pages available in the Internet is a Groupware system. Discuss this view. If that is true, what are the main aspects of this system? Describe the model underlying this of these aspects? Is it an elaborate model or a trivial one?
8. *[Level 3]* In some places in this chapter we mentioned virtual reality. Investigate if there are groupware applications that use virtual reality as a way

of interfacing with the user. Doesn't current technology VR has some of the problems discussed in using WWW as the underlying infrastructure of groupware? Is VRML, for example, an appropriate technology for non-client-server applications?

9. *[Level 3]* In engineering design there is a strong movement towards "Concurrent Engineering," which state that many specialists in areas such as as marketing, manufacturing, cost, materials, maintenance, and so on, must take part in the design of engineering artifacts. Investigate what groupware tools are being used by concurrent engineering practitioners, specially what multi-user support does CAD systems provide.

10. *[Level 4]* In the writing of this chapter, the authors did not use very elaborated groupware. We divided the chapter into sections and for each section one of us had the writing role, while the other had the reviewing and commenting role. In the beginning we would take turns, once a section was written the author would wait for the reviewers changes and comments before working on that section again. By the end of the writing the interaction became more intense and there times when an author would make changes in a section before receiving the comments, and thus the comments and changes would be out-of-date. We use e-mail to send the LaTeX file from one to the other, that is we used a communicator as the means of our collaboration. Specify a keeper that would be appropriate for the task and for our mode of working. Take into consideration that each of us have different preferences as to which text editor to use, that Internet connection between us was slow and unreliable. The keeper should be appropriate for both the turn-taking and the "closer-to-deadline" working patterns.

11. *[Level 4]* Design a group spreadsheet in which the four aspects described in this chapter are carefully taken into account and articulated. Create a design specification that includes description of spreadsheet operations, multi-user features, group user interface, centralized versus distributed implementation, access control, and concurrency control. Emphasis of your design should be on the groupware aspects and features that allow a group to all use the spreadsheet at the same time, and also at different times. Do not spend much effort on the single user features that are identical to those of single user spreadsheets. State clearly any assumptions or extensions that you make; feel free to suggest creative ideas and innovative designs.

10.11 References

1. P. Antunes, N. Guimaraes, J. Segovia, and J. Cardenosa. Beyond formal processes: Augmenting workflow with group interaction techniques. In *Conference on Organizational Computing Systems*, pages 1–9. ACM Press, 1995.
2. R. F. Bales and S. P. Cohen. *SYMLOG: A System for Multiple Level Observation of Groups*. The Free Press, 1979.
3. S. D. Benford and L. E. Fahlen. A spatial model of interaction in virtual environments. In *Proceedings of The Third European Conference on Computer Supported Cooperative Work (ECSCW'93)*, pages 109–124, 1993.
4. R. Bentley, U. Busbach, D. Kerr, and K. Sikkel, editors. *Computer Supported Cooeprative Work: The Journal of Collaborative Computing*, volume 6. Kluwer Academic Press, 1997. Special issue on Groupware and the World Wide Web.
5. R. Bentley, J. A. Hughes, D. Randall, T. Rodden, P. Sawyer, D. Shapiro, and I. Sommerville. Ethnographically-informed systems design for air traffic control. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'92)*, pages 123–129, Toronto, Ontario, 1992. ACM Press.
6. S. Bharwani. The MIT design studio of the future. Videotape presentation at the ACM CSCW'96 Conference, 1996.
7. K. Birman. The process group approach to reliable distributed computing. *Communications of the ACM*, 36(12):37–53, 1993.
8. R. Braham and R. Comerford. Sharing virtual worlds. In *IEEE Spectrum*, volume 34, pages 18–20, 1997.
9. BSCS. http://bscw.gmd.de/.
10. S. Chengzheng et al. A generic operation transformation scheme for cooperative editing systems. In *Proceedings of the 1997 ACM SIGGROUP Conference on Supporting Group Work*, pages 425–434. ACM Press, 1997.
11. J. Conklin and M. Begeman. gIBIS: An hypertext tool for exploring policy discussion. In *Proceedings of the Second Conference on Computer Supported Cooperative Work*, Portland, OR, September 26–28 1998.
12. W. Croft and L. Lefkowitz. Using a planner to support office work. In *Proceedings of the ACM COIS*, pages 55–62. ACM Press, 1988.
13. CTRG. http://www.cs.colorado.edu/s̃kip/ctrg.html.
14. CU-SeeMe. http://cu-seeme.cornell.edu/W̃CW/.
15. J. Dietz et al. Speech acts or communicative action? In *Proceedings of The 1991 European Conference on Computer Supported Cooperative Work*, pages 235–248, 1991.
16. C. Ellis, S. J. Gibbs, and G. Rein. Design and use of a group editor. In G. Cockton, editor, *Engineering for Human Computer Interaction*. North Holland, 1990.
17. C. Ellis, S. J. Gibbs, and G. Rein. Groupware: Some issues and experiences. *Communications of the ACM*, 34(1):38–58, 1991.
18. C. Ellis and C. Maltzahn. Chautauqua: Merging workflow and groupware. In *Proceedings of the HICSS'96 Conference (Hawaii International Conference on Systems Science)*, 1996.
19. C. Fernstrom. Process Weaver: Adding process support to UNIX. In *2nd*

International Conference on the Software Process, pages 12–26. IEEE Computer Society Press, 1993.

20. G. Fischer, K. Nakakoji, J. Oswald, G. Stahl, and T. Sumner. Embedding computer-based critics in the context of design. In *Human Factors in Computing Systems, INTERCHI'93 Conference Proceedings*, pages 157–164. ACM, 1993.
21. F. Flores, M. Graves, B. Hartfield, and T. Winograd. Computer systems and the design of organizational interaction. *ACM Trans. Office Information Systems*, 6(2):153–172, 1988.
22. R. Grinter. Using a configuration management tool to coordinate software development. In *Proceedings of the 1995 ACM SIGOIS Conference on Organizational Computing Systems*, pages 168–177. ACM Press, 1995.
23. J. Grudin. Groupware and social dynamics: eight challenges for developers. *Communications of the ACM*, 37(1):92–105, 1994.
24. W. F. Heitman. The Business Improvement Lab: 1997 Summary of Best Practices, 1997.
25. C. Hwang and M. Lin. *Group Descision Making under Multiple Criteria: Methods and Applications*. Srpinger Verlag, 1987.
26. id Software. http://ww.idsoftware.com/.
27. INOTE. http://jefferson.village.virginia.edu/m̃ar4g/index.html.
28. H. Ishii et al. Clearface: Translucent multi-user interface for teamworkstation. In *Proceedings of The 1991 European Conference on Computer Supported Cooperative Work*, pages 163–174, 1991.
29. IVS. http://www.inria.fr/rodeo/personnel/Thierry.Turletti/ivs.html.
30. S. Jablonski and C. Bussler. *Workflow Management Systems: Modeling, Architecture, and Implementation*. International Thomson Computer Press, 1996.
31. R. Johansen. *Groupware: Computer Support for Business Teams*. The Free Press, New York, 1988.
32. R. Johansen. *Leading Buisness Teams*. Addison-Wesley, Reading, Mass., 1991.
33. D. Karagianis. The adonis workflow system. In *Proceedings of the 1996 Linz Workshop on Workflow Systems,*, Linz, Austria, 1996.
34. V. Kumar. *MBone: Interactive Multimedia On The Internet*. Macmillan Publishing, 1995.
35. W. Kunz and H. Rittel. Issues as elements of information systems. Technical report, Working paper No. 131, Institute of Urban and Regional Development, Univ. of California, Berkeley, 1970.
36. W. Lehnet. *Internte 101*. Addison-Wesley, 1998.
37. H. Linstone and M. Turoff. *The Delphi Method: Techniques and Applications*. Addison-Wsley Publishing Co., 1975.
38. M. Losada and S. Markovitch. Groupanalyzer: A system for dynamic analysis of group interaction. In *Proceedings of the 23rd Hawaii International Conference on Systems Science*, 1990.
39. Lotus. Lotus notes. http://www.lotus.com.
40. Mbone. http://www.mbone.com/.
41. J. H. Morris. Issues in the design of computer support for co-authoring and commenting. In *Proceedings of ACM CSCW'90*, pages 183–195, 1990.

42. C. M. Neuwirt, D. S. Kaufer, R. Chandhok, and J. H. Morris. Issues in the design of computer support for co-authoring and commenting. In *Proceedings of the Third Conference on Computer Supported Cooperative Work*, pages 183–195. ACM, Los Angeles, CA October 7–10 1990.

43. J. Nunamaker, A. Dennis, J. Valacich, D. Vogel, and J. George. Electonic meeting systems to support group work. *Communications of the ACM*, 34(7), 1991.

44. QuestMap. http://www.cmsi.com/business/info/.

45. RFC2109. HTTP state management mechanism. http://www.cis.ohio-state.edu/htbin/rfc/rfc2109.html.

46. M. Roseman and S. Greenberg. Teamrooms: Network places for collaboration. In *Proceedings of the 1996 ACM CSCW Conference*, pages 325–333. ACM Press, 1996.

47. P. Sachs. Transforming work: Collaboration, learning and design. *Communications of the ACM*, 38(9):36–44, 1995.

48. J. R. Searle. *Speech Acts*. Cambridge University Press, 1969.

49. L. Suchman. Do categories have politics? the language/action perspective reconsidered. In *Proceedings of the Third European Conference on Computer-Supported Cooperative Work*, pages 1–14, 1993.

50. ON Technologies. Meetingmaker. http://www.on.com/mmxp/.

51. M. Turoff. Computer-mediated communication requirements for group support. *Journal of Organizational Computing*, 1(1):85–113, 1991.

52. T. Winograd and F. Flores. *Understanding Computers and Cognition*. Addison Wesley, 1987.

53. XTV. http://www.visc.vt.edu/succeed/xtv.html.