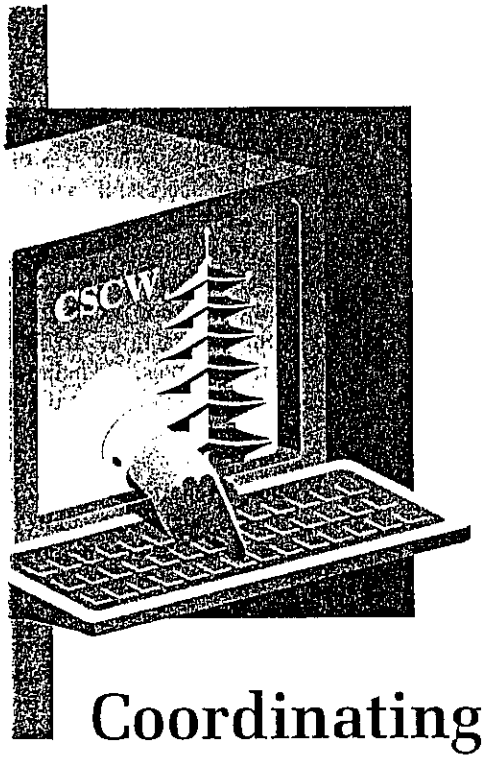


When Client/Server Isn't Enough:

Coordinating Multiple Distributed Tasks

Ting-Peng Liang, Hsiangchu Lai, and Nian-Shing Chen
National Sun Yat-sen University
Hungshiung Wei, Billion Electric Company
Meng Chang Chen, Academia Sinica

Because group multitasking involves interactions within and across applications, this article suggests a three-tiered architecture that includes a groupware server, application servers, and clients.

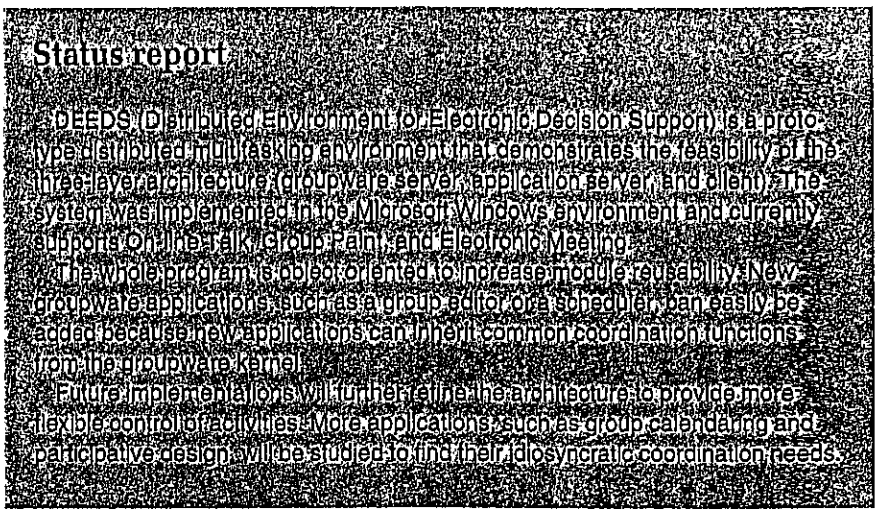
The rapid growth of computer networks and client/server computing has focused much attention on the development of groupware — computer software that supports cooperative work. In the past several years, prototypes and commercial products have been developed in many areas, such as group writing (a group of writers simultaneously writing and editing a document), electronic meetings (group participants discussing issues or making decisions electronically), workflow management (managing electronic documents and dataflows among agents), and group scheduling (scheduling group activities based on group members' individual timetables).[1-3] Groupware differs from traditional software in that it allows a group of people to work together electronically, requires a distributed hardware environment (networked personal computers or workstations), and utilizes client/server computing to facilitate information sharing, dissemination, routing, and user interaction.

However, the traditional two-tiered client/server architecture for distributed computing is inadequate in a multitasking group environment because coordination in this case must take into consideration not only the interactions within an application, but also the interactions across multiple applications. When designing such systems, therefore, we must differentiate the coordination functions specific to a particular application from those shared by all applications. In the following, we present a three-layer architecture that extends the client/server architecture to include a groupware server, application servers, and application clients.

Groupware coordination

As currently conceived, a groupware system includes two major modules: (1) a computation module that processes individual activities, and (2) a coordination module that "binds separate activities into an ensemble."[4] Since group work involves multiple agents conducting different activities concurrently, task coordination has a critical impact on group productivity.

0018-9162/94/$4.00 © 1994 IEEE



The meaning of "coordination" may vary in different situations. Broadly speaking, it means "the integration and harmonious adjustment of individual work efforts toward the accomplishment of a larger goal."[1] This interpretation can be further decomposed into more detailed issues such as how can the overall goal be divided into actions? How can actions be assigned to groups or to individual actors? How can resources be allocated among different actors? How can information be shared among different actors to achieve the overall goal?[5]

In designing computer-based systems for cooperative work, coordination refers to the management of interdependencies and interactions among some set of agents performing a collective activity through computers.[6] Here, "management" includes creation, communication, synchronization, and control of processes and activities. The complexity and importance of coordination are functions of system types and group size.

Groupware can be classified into four categories based on two dimensions: synchronous or asynchronous and single-tasking or multitasking. Single-tasking systems let a group work on a single task at a time, whereas multitasking systems allow participants to work on different tasks. In general, coordination requirements and complexity are higher for synchronous and multitasking applications.

In this article, we examine the problems involved in coordinating group activities and present a useful mechanism for coordinating multiple tasks in a group computing environment. We start by identifying the key components and coordination needs of various types of groupware. We then propose a three-layer architecture for coordinating multiple tasks in a group decision environment. Finally, we present a prototype implementation of our DEEDS (Distributed Environment for Electronic Decision Support) coordination mechanism. The system currently supports three group applications (On-line Talk, Group Paint, and Electronic Meeting) that can be used simultaneously in a Microsoft Windows-based environment.

Coordination requirements

In a group computing environment, the many possible cooperative endeavors make it necessary to coordinate all activities, including message direction, process control, consistency checking, conflict resolution, and version control. Previous research has identified certain coordination requirements in different contexts. For instance, Greif and Sarin discussed requirements for coordinated use of databases in groupware.[7] Harrison, Ossher, and Sweeney emphasized the importance of maintaining consistency in concurrent development.[8] Lun and MacLeod claimed that real-time interactive systems must support agent dialogue and agent interaction.[9] Agent dialogue includes queries, requests, and complaints that let users share messages to increase their knowledge. Agent interaction includes negotiation, synchronization, and choices that support ongoing concurrent user activities. Patterson and Meeks suggested that a synchronous multiuser application must provide shared flexible control, sound session management, and successful performance of activities.[10]

The complexity of coordination is affected by several factors including the nature of applications, the working environment, agent interaction, artifact control, the nature of tools, and maintenance requirements. Let's look at these coordination issues individually.

Nature of application. Different groupware applications have different coordination needs and can be categorized in several ways. The most significant factor in coordination complexity is whether the application requires real-time interaction. For example, a synchronous, real-time conversation system is more difficult to coordinate than an asynchronous workflow system sending electronic documents based on predetermined flow models, because the former has to handle complicated concurrency control and conflict resolution.

We can also classify groupware applications by their major functions: communication, collaboration, and group decision making. The core function of communication-oriented applications, such as on-line talk or workflow management, is to provide a channel through which messages can be passed correctly. Communication coordination requirements, as shown in Table 1, include the proper handling of message routing, activity sequencing, and access control.

Collaborative work, such as jointly designing a diagram or writing a paper, is different from communication-oriented tasks in that a group of users work cooperatively on a tangible object to complete a task. In group writing, for example, each user may write part of an article, but the combination of all users' work becomes a complete paper. Coordination in this case requires version control, consistency checking, and concurrency control, in addition to the aforementioned message routing, activity sequencing, and access control.

Decision-oriented groupware, or group decision support systems (GDSS), supports decision making by a group of people working together to choose among alternatives. The most well-known GDSS is the electronic meeting system developed at the University of Arizona.[3] Group decision-making systems differ from communication systems in that they support idea generation and

alternative selection. They also differ from collaboration in that the result generated from the group may not be the direct combination of individual works. For instance, only one alternative may eventually be chosen by the group, even if system users have generated more than a hundred. The coordination requirements are similar for GDSS and collaboration systems, though their actual mechanisms and complexities may differ.

Working environment. Two aspects of the working environment affect groupware coordination: (1) whether it is a single-tasking or a multitasking environment, and (2) whether individual control of the working environment is allowed. In a single-tasking environment, such as an electronic meeting system or a workflow system, the coordination mechanism must take care of the requirements specific to the application only. In other words, coordination is limited to activities within a particular application.

In a multitasking environment, the system must support the multiple requirements of different applications. For instance, DEEDS supports three different types of applications: Talk (a communication-based application), Group Paint (a collaboration-oriented application), and Electronic Meeting (a group decision-making system). Each participant can use any of these applications at any time. Coordination involves not only activity coordination within an application, but also activity coordination across application borders.

The second coordination issue is to what extent users can customize their working environments. A tailored environment should give users a number of options, ranging from selection of window colors to choice of computing platform. An environment allowing several computing platforms to work together definitely needs a more complex coordination mechanism than one that only lets the user change screen colors.

Agent interaction. Since coordination involves harmonizing multiple activities, the way agents interact also affects requirements. In the multitasking group environment, every participant can join several groups having many participants. Therefore, agent interaction comprises five different types: (1) individual to individual, (2) individual to group, (3) individual to all, (4) group to group, and (5) group to all. Here, *group* refers to the participants working on the same task (such as drawing a graph), whereas *all* means all users signing onto the groupware environment (which may include more than one group).

In addition to interaction type, we must consider participant relationship. In a group decision environment, at least two kinds of relationships exist: hierarchical and peer-to-peer. The major feature of a hierarchical group is that certain members, such as the classroom instructor or the company manager, have higher authority than others in the group. Coordination mechanisms in this case must differentiate supervisors from subordinates and give them different priorities. The peer-to-peer group assumes that all participants are equal.

The third aspect of agent interaction concerns the relationship between participants and the application program. The coordination mechanism must be able to resolve certain questions: (1) Can a participant join or withdraw freely? (2) Is there an upper limit on the number of participants? (3) Can a participant be forced out? and (4) Are there time or other constraints on system usage?

Artifact control. Artifacts are objects upon which the participants work. In group writing, for instance, the document coauthored by participants is the artifact. The coordination issues associated with artifacts include artifact management, user privilege management, consistency control, modification propagation, and concurrency control. They are briefly described in Table 2.

Application tools. Different applications may use different tools that require different coordination mechanisms. For example, it may be necessary to lock data records so that a particular alternative can be edited when the brainstorming tool is used in an electronic meeting system. This becomes unnecessary when the voting tool is used because users do not need to access the same file or document. The following three relationships are critical to the coordination requirements associated with application tools:

(1) Tool to participants: Who can use which tools?
(2) Tool to working environment: What tools are appropriate for applications in a certain environment? How can tools be adapted to a different environment?
(3) Tool to artifact: Which tools have been used to work on what artifacts?

Table 1. Coordination requirements of different applications.

Type of Applications	Coordination Needs
Communication	Routing, sequencing, access control
Collaboration and Group decision making	Routing, sequencing, access control, version control, consistency checking, concurrency control

Table 2. Coordination issues associated with artifacts.

Issue	Description
Privilege management	Whether a user can create, modify, or delete an artifact.
Artifact management	How to define, store, manipulate (such as overlapping artifacts), and retrieve artifacts.
Consistency control	How to ensure that all users receive the same version.
Modification propagation	Where and how soon modifications should be propagated.
Concurrency control	Locking or other mechanisms for concurrency control.

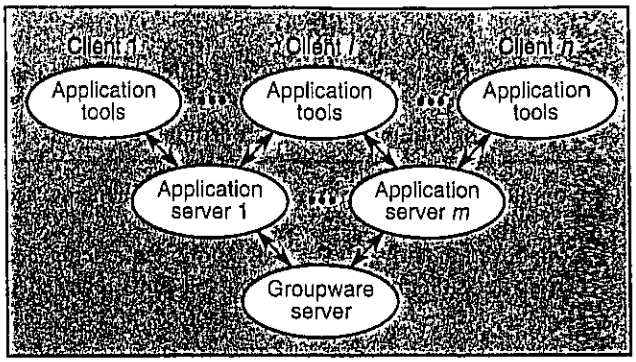


Figure 1. The traditional two-layer client-server architecture for distributed computing is inadequate in a multitasking group environment. An extension to three layers, as shown right, is more feasible.

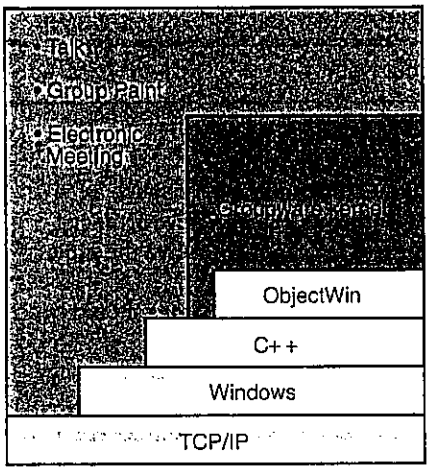


Figure 2. The DEEDS development environment allows for the addition of new groupware applications because new applications can inherit common coordination functions from the groupware kernel.

System maintenance. System maintenance refers to operations required for smooth operation of the system and includes data recovery, data back-up, and so on. It is also important that data accuracy and currency be maintained while the system interacts with a group of participants.

A three-layer architecture for task coordination

The traditional client/server architecture for distributed computing includes two layers: a client layer and a server layer. System functions specific to the user application, such as user interface modules, are allocated to the client, whereas functions involving user interaction or general-to-multiple applications are allocated to the server. This is inadequate in a multitasking group environment because coordination in this case must take into consideration not only interactions within an application, but also interactions across multiple applications. When designing such systems, therefore, we must differentiate the coordination functions specific to a particular application from those shared by all applications. In this section, we present a three-layer architecture that extends the client/server architecture to three levels: a groupware server, application servers, and application clients. Figure 1 illustrates their relationships.

Groupware server. The groupware server's primary objective is to provide the coordination functions necessary for all groupware applications. It maintains the basic communication functions such as message routing and sequencing, as well as user management functions such as adding or deleting users. For message routing, the server keeps track of which application is running and the location of each application server and its clients so that messages can be delivered to the correct destination. For user management, the groupware server maintains user profiles and privileges for all applications. It also controls access to applications. The major advantage of adding this level is that it takes care of common, domain-independent functions, which allows more flexibility to the other two levels in handling domain-specific coordination needs.

When a user signs on to the system, the groupware server checks whether the user is authorized to enter and, if so, which applications the user can access. If the user asks to join an ongoing electronic meeting, the groupware server checks whether the user has that privilege and then informs the electronic meeting server (an application server) of the user's request. The meeting server then decides whether the user is acceptable, and this decision is passed on to the user.

Application server. An application server is a system that takes care of coordination and other needs associated with the execution of a particular application. It is built on top of the groupware server. Each application in a multitasking environment must have an application server. For example, a group painting system may need a mechanism that partitions the painting area into several subareas and then integrates individual users' contributions into a complete work. Since this partition mechanism is unique to Group Paint, it is more appropriately implemented and maintained by the Group Paint application server. Similarly, a polling mechanism that allows ballots to be disseminated and collected is suitable for the Electronic Meeting server but may not be useful for workflow or Group Paint. The major responsibilities of an application server include

(1) controlling access to application tools,
(2) performing version control,
(3) checking consistency to ensure that changes are properly recorded and propagated, and
(4) controlling concurrency and resolving conflicts in real-time applications.

Client. The layer on top of the application server is called the client. Its main role is to provide user interfaces and application tools so that various activities can be performed. For example, Group Paint needs drawing tools such as pen, brush, shapes, and eraser. All of these are provided at the client layer. The voting and group decision tools are sample client functions of the Electronic Meeting system.

DEEDS: A prototype

DEEDS is a prototype distributed multitasking environment that demonstrates the feasibility of the three-layer architecture. The system was implemented in the Microsoft Windows environment, using Borland C++ and the WinSocket library as development tools. Since applications can be classified into communication, collaboration, and group decision making, we chose one typical application from each category: On-line Talk for communication, Group Paint for collaboration, and Electronic Meeting for group decision making. Figure 2 illustrates the development environment of DEEDS.

As shown in Figure 2, DEEDS uses the TCP/IP protocol for network communication. TCP/IP was chosen for two reasons: (1) It supports communication on

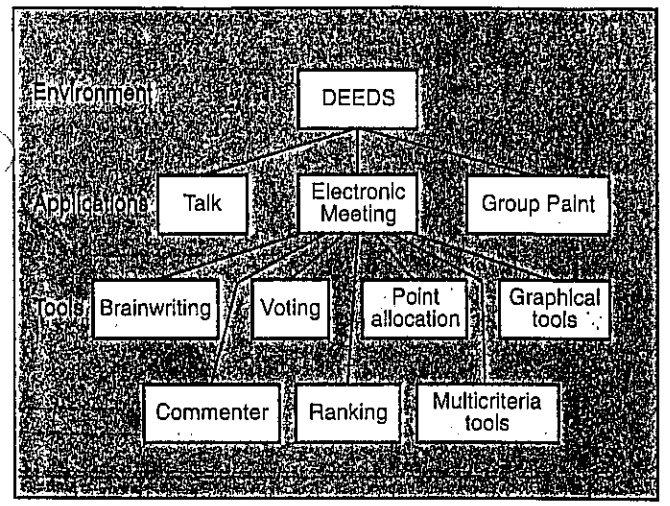


Figure 3. Three subsystems constitute the application functions of DEEDS. One of them, Electronic Meeting, provides tools for group decision making.

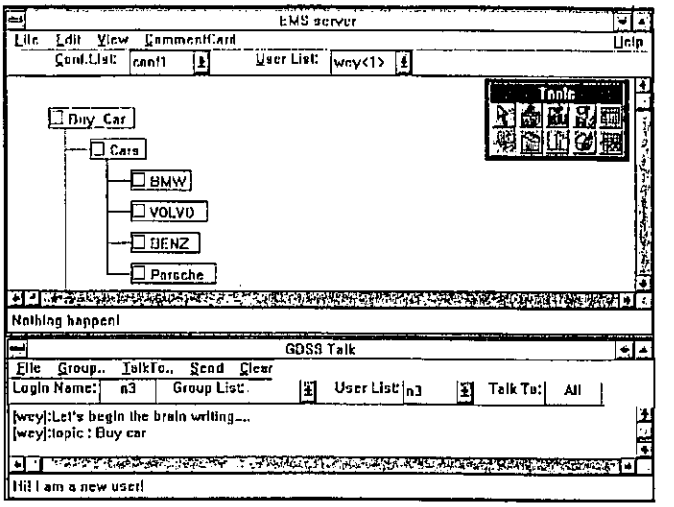


Figure 4. During an idea generation session, participants can enter ideas into any node of the tree.

both local area networks and wide area networks, and (2) it is a standard not bound to a particular vendor. On top of TCP/IP, the application system is built in the Microsoft Windows environment. Windows was chosen primarily for its popularity and its support of multiple windows.

The application system comprises a groupware kernel and several application modules. The groupware kernel includes the necessary functions for communication and session control. The application modules include the application-specific functions of Talk, Group Paint, and Electronic Meeting. The whole program is object oriented to increase module reusability. We can easily add new groupware applications, such as a group editor or a scheduler, because new applications can inherit common coordination functions from the groupware kernel. Programmers need only write the application-specific portion.

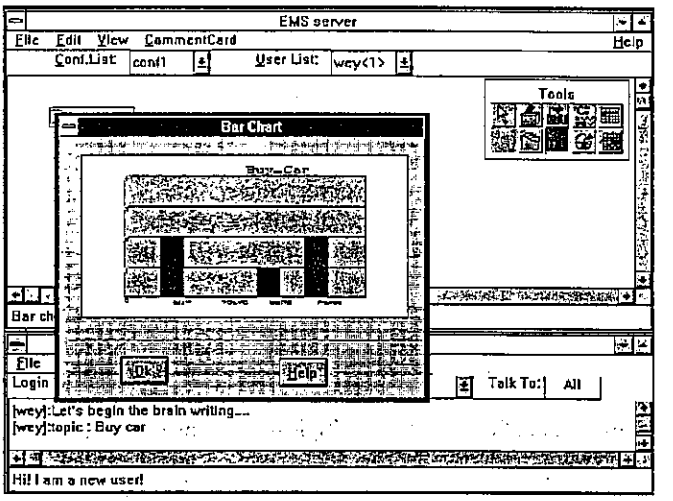


Figure 5. After ideas have been voted on, results can be displayed in tables, bar charts, or pie charts.

Application functions. The application functions of DEEDS belong to its three subsystems: Talk, Group Paint, and Electronic Meeting. Figure 3 shows the major components.

Talk is an on-line communication tool that lets users exchange messages with individual users, a user group, or all system users. This provides an ad hoc communication channel for meeting participants or group designers using other tools.

Group Paint is the group version of the popular Paintbrush program. It includes regular Paintbrush functions such as pen, brush, eraser, spray, alphabetical letters, line drawing, and shape drawings such as circles and squares. But in addition to individual usage, Group Paint allows a group of users to draw cooperatively.

Electronic Meeting consists of tools for idea generation, idea evaluation, and result presentation. We use brainwriting as an idea generation tool through which a group of participants can enter their ideas. Every participant sees his/her own ideas as well as the ideas entered by other group members. All ideas are organized into a tree to show their relationships.

Figure 4 shows the result of a sample idea generation session for purchasing a car. Four alternatives have been proposed: BMW, Volvo, Mercedes Benz, and Porsche. In the idea generation session, participants can enter their ideas into any node of the tree. They may also move nodes and expand the tree. The bottom third of the screen shows a Talk session running during the meeting. The chair of the electronic meeting uses Talk to inform participants that the discussion is about buying a car.

Once enough ideas have been generated, the meeting chair can stop the brainwriting and request idea evaluation and decision making. DEEDS's idea evaluation tools include idea commenting, voting, ranking, point allocation, and multicriteria decision tools (Figure 3). The chair chooses a tool and drags the tool icon from the toolbox to the alternatives just generated. If the chair activates the voting tool, for example, ballots will be created and electronically mailed to all eligible participants. Participants vote for the alternatives they support; the results are sent to the meeting chair, tabulated by the voting tool, and then displayed in tables, bar charts, or pie charts. Figure 5 shows sample voting results displayed in a bar chart.

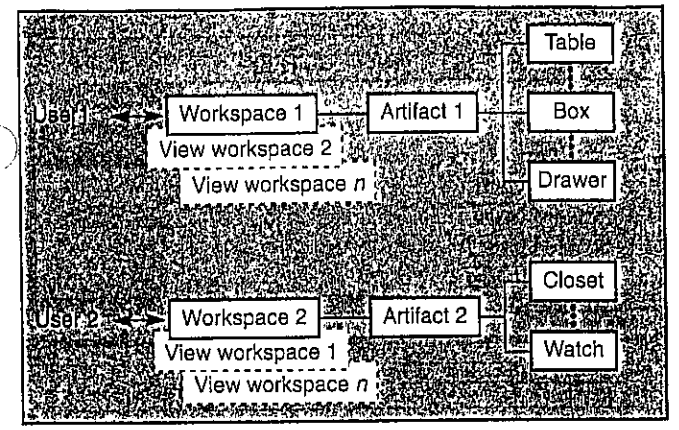


Figure 6. In Group Paint, each user works in a separate workspace but can view the workspaces of others.

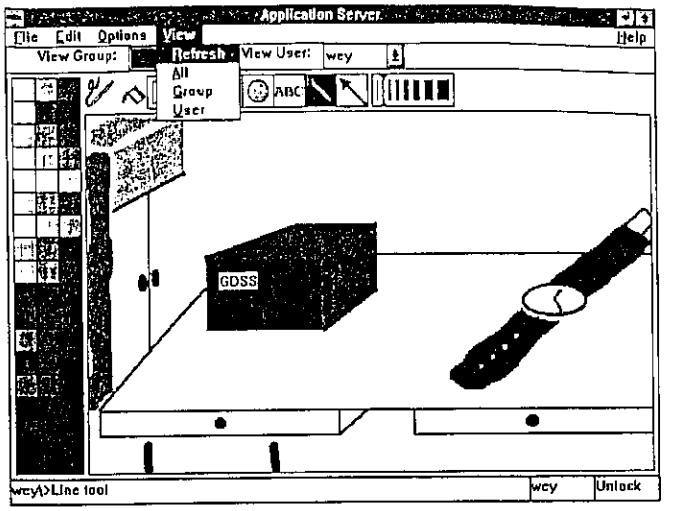


Figure 7. The results of activity in the two user workspaces of Figure 6 are combined in a screen overlay.

Task coordination. Task coordination in DEEDS follows the three-layer architecture. The groupware server conducts most of the interapplication coordination. We defined application protocols to identify each application so that the groupware server knows the applications with which a particular message is affiliated.

Intra-application coordination is conducted by the application server. Since different applications have different coordination needs, application server complexity also varies. For example, Talk offers tools only for receiving and sending text, and the primary function of its server is sequencing all messages properly.

Coordination in Group Paint is more complicated. For instance, multiple users may want to work on the same area, which would result in a conflict. A user may want to delete an object created by another user or add an object on top of an existing object created by someone else. In other words, the application server must manage space allocation, object ownership, and user privilege.

Group Paint uses a workspace-based approach to handle coordination at the application server. Simply put, the application server assigns an empty workspace to every user who signs on to the system. The user has full rights of the workspace, including read, write, delete, and modify. The user can view the workspaces of other participants but cannot write, delete, or modify objects in those workspaces without authorization from the owner. This reduces the coordination complexity.

When the results of a joint effort need to be seen, a user can combine all users' work flexibly. Workspaces can be overlaid to show the combined effect. For example, Figure 6 indicates that two users are using Group Paint to design a picture. User 1 draws a table, a box, and a drawer in workspace one. User 2 draws a closet and a watch in workspace two. Figure 7 shows the resulting screen overlay combining these two workspaces.

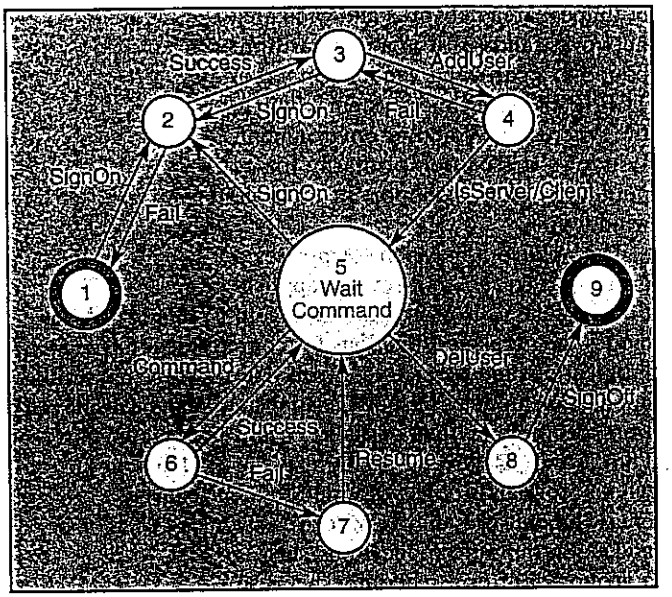


Figure 8. A series of state transitions occur at the client site when a user joins or initiates a conference. After a user is recognized and accepted, the client system enters states 5, 6, and 7 to process activity commands.

The coordination in Electronic Meeting is the most complicated of the three applications. Sometimes, human interruption may be necessary. For example, when the electronic meeting moves from one stage into another, the chair needs to communicate with the participants to ensure that they follow the meeting. When enough ideas have been collected and a vote is requested, the chair must stop idea generation and initiate voting. These stage-switching activities must be handled by humans.

Since most coordination needs are handled by the servers, the design of client programs becomes simple in DEEDS. Figure 8 shows the state transition diagram of the activities performed at the client site. First, the user requests to sign on to DEEDS. This brings the system to state 2, where it waits for a response from the groupware server. If the user is legal, the system enters state 3 to choose applications. The user may ask to initiate a new conference or to join an existing conference. This request initiates state 4. The groupware server checks whether it is legal for the user to create a new conference or join an existing one. If it is, the client system enters states 5, 6, and 7 to process activity commands. After finishing the work, the user withdraws from the conference. This brings the user

to state 8, which is essentially the same as state 2, allowing the user to join a new application or leave the system.

The computing environment has evolved from single-tasking to multitasking and from stand-alone machines to networked machines. Computing in a distributed environment for multitasked cooperative work is a promising area that presents many coordination issues. Our prototype system implements a three-layer architecture to provide greater control and flexibility in the distributed multitasking environment. The architecture can be further refined to provide more flexible control of activities. More applications, such as group calendaring and participative design can also be studied to find their idiosyncratic coordination needs and to elaborate the division of labor among different servers and clients. ■

Acknowledgments

This research was supported in part by the Technology Research Division of the Institute for Information Industry (TRD-82-021) and the National Science Council of the Republic of China (NSC83-0301-H-110-002). We thank the following graduate students at the National Sun Yat-sen University for their assistance in implementation: Tian-Lian Chen, Shiao-Hong Jin, Guo-Shu Hong, Ming-Yung Hong, Bo-chang Hsieh, Bo-Hu Hsu, and Tian-You Hwang.

References

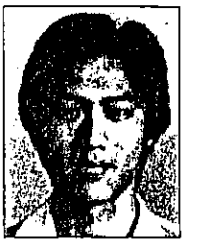
1. C.A. Ellis, S.J. Gibbs, and G.L. Rein, "Groupware: Some Issues and Experiences," *Comm. ACM*, Vol. 34, No. 1, Jan. 1991, pp. 38-58.
2. *Computer-Supported Cooperative Work and Groupware*. S. Greenberg, ed., Academic Press Ltd., London, 1991.
3. J.F. Nunamaker et al., "Electronic Meeting Systems to Support Group Work," *Comm. ACM*, Vol. 34, No. 7, July 1991, pp. 40-61.
4. D. Gelernter and N. Carriero, "Coordination Languages and Their Significance," *Comm. ACM*, Vol. 35, No. 2, Feb. 1992, pp. 97-107.
5. T.W. Malone and K. Growston, "What is Coordination Theory and How Can It Help Design Cooperative Work Systems?" *Proc. CSCW 90*, ACM Press, New York, 1990, pp. 357-370.
6. W.R. Zhang et al., "A Cognitive-Map-Based Approach to the Coordination of Distributed Cooperative Agents," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 22, No. 1, Jan-Feb. 1992, pp. 103-114.
7. I. Greif and S. Sarin, "Data Sharing in Group Work," *ACM Trans. Office Information Systems*, Vol. 5, No. 2, Apr. 1987, pp. 187-211.
8. W.H. Harrison, H. Ossher, and P.F. Sweeney, "Coordinating Concurrent Development," *Proc. CSCW 90*, ACM Press, New York, 1990, pp. 157-167.
9. V. Lun and I.M. MacLeod, "Strategies for Real-Time Dialogue and Interaction in Multiagent Systems," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 22, No. 4, July-Aug. 1992, pp. 671-680.
10. J.F. Patterson and W.S. Meeks, "Rendezvous: An Architecture for Synchronous Multiuser Applications," *Proc. CSCW 90*, ACM Press, New York, 1990, pp. 317-327.

Ting-Peng Liang is director of the Institute of Information Management of the National Sun Yat-sen University. His primary research interests include groupware, decision support systems, and expert systems.

Liang received his PhD degree in information systems from the University of Pennsylvania. He is a member of the IEEE Computer Society.

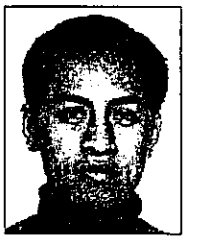
Hsiangchu Lai is an associate professor in the Department of Information Management, National Sun Yat-sen University and a visiting scholar at the University of Texas at Austin. Her research interests include negotiation support systems, group decision support systems, and strategic information technology.

Lai received her PhD in management information systems from Purdue University. She is a member of the IEEE Computer Society, ACM, and TIMS/ORSA.



Nian-Shing Chen is an associate professor in the Department of Information Management and director of the Computer Network Division of the Computer Center of National Sun Yat-sen University. His research interests include computer networks, distributed computing, and computer-supported cooperative work.

He received his doctoral degree from National Tsing Hua University and is a member of IEEE.



Hungshiung Wei is a researcher at Billion Electric Co. in Taipei, Taiwan. His research interests include ISDN, computer networks, and decision support systems.

He received his master's degree in information management from National Sun Yat-sen University, Kaohsiung, Taiwan.

Meng Chang Chen is an assistant research fellow at the Institute of Information Science, Academia Sinica, Taiwan. His research interests include database systems, groupware, distributed systems and processing, and knowledge discovery and representation.

He received a PhD in computer science from the University of California at Los Angeles. He is a member of ACM.

Readers can contact the authors through Ting-Peng Liang, Institute of Information Management, National Sun Yat-sen University, Kaohsiung, Taiwan; e-mail liang@mis.nsysu.edu.tw

ADVERTISER INDEX

*International edition
**U.S. edition

FOR DISPLAY ADVERTISING INFORMATION, CONTACT:

Southern California and Mountain States: Richard C. Faust, 24050 Madison Street. Suite 101, Torrance, California 90505; Tel: (310) 373-9604; Fax: (310) 373-8760.

Northern California and Pacific NW: William W. Hague, 9017 Peacock Hill Road, Gig Harbor, Washington 98332; Tel: (206) 858-7575; Fax: (206) 858-7576.

East Coast: Gail A. Frank, Nancy Inserra, Art Frank, 82 Bethany Road, Suite 4, Hazlet, New Jersey 07730; Tel: (908) 264-1100; Fax: (908) 264-4340.

New England: Paul Gillespie, PO Box 6444, Holliston, Massachusetts 01746; Tel: (508) 429-8907; Fax: (508) 429-8684.

Southwest: Frank E. Johnson, 3601 Smith-Barry Road, Suite 103, Arlington, Texas 76013; Tel: (817) 275-2651; Metro Voice/Fax: (817) 265-3811.

Southeast: Megan Wendling, 460 29th Street NW, Naples, Florida 33940; Tel: (813) 353-6412; Fax: (813) 455-0031.

Midwest: Harold L. Leddy, 345 Auburn Avenue, Winnetka, Illinois 60093-3603; Tel: (708) 446-8764; Fax: (708) 446-7985.

For production information, conference, and classified advertising, contact Marian B. Tibayan, *Computer*, 10662 Los Vaqueros Circle, Los Alamitos, California 90720-1264; e-mail: mtibayan@computer.org; Tel: (714) 821-8380; Fax: (714) 821-4010.

PRODUCT INDEX