Multimedia Tools and Applications, 9, 199–226 (1999)

A Multimedia World Wide Web Based Conference Minute System for Group Collaboration

ING-CHAU CHANG
Department of Information Management, ChaoYang University of Technology, WuFeng, Taichung County, Taiwan

BO-SHEN LIOU
JAU-HSIUNG HUANG jau@cmlab.csie.ntu.edu.tw
SHIUH-SHENG YU AND CHEE-WEN SHIAH
Communications and Multimedia Laboratory, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

Abstract. In this paper, we propose a World Wide Web based conference minute system which preserves important information in the conference as web documents and maintains hyperlinks to related minutes to accelerate the progress of the project. This system, based on our Java application sharing framework, provides capabilities to record and replay the audio/video data of the video conference and operations of the shared applications used in the meeting in a synchronous manner. Through the hyperlinks, people can quickly understand the logic flow of every meeting held in the project duration and select any part in a meeting for replay to know what has actually happened. Furthermore, when a new meeting is held, a minute template is generated by the system to inherit the hyperlinks from its predecessors. Other related minutes and the execution status of the resolutions in the project can be easily accessed through the hyperlinks between them. Through the help of the system, execution of the project and management of its organizational memory could be easily achieved to increase the productivity of the collaborative groups with geographically dispersed members.

Keywords: computer supported cooperative work (CSCW), multimedia, world wide web (WWW), conference minute, organizational memory

1. Introduction

1.1. Motivation

With the advances of computer and communication technologies in the past several years, people geographically dispersed at different locations work more and more closely related by using tools and systems developed within the field of Computer Supported Cooperative Work (CSCW); it is also referred as Groupware. The CSCW system refers to groups of people who work for a common goal and seek to discover how technologies can help them [13]. This common goal could be a simple co-editing process for a document, or a complex large-scale construction project lasting for years. To achieve the goal, lots of CSCW systems, such as video conference systems [6, 8, 15], shared whiteboards [6, 8, 15], application-sharing systems [1, 3, 10, 11, 14, 23, 24, 26, 38], scheduling systems [6], workflow systems [30, 45], etc., were developed to help people in their collaborative works in a time efficient way.

Collaborative projects with geographically dispersed participants involved are ordinarily composed of series of planning, designing, execution and feedback processes, which are closely linked by meetings. As the progresses of these meetings, issues are first raised, then corresponding assertions are proposed by group participants. After thorough discussion about the issues, resolutions may be reached with deliberation and waited for execution. These resolutions must be carefully executed after the meeting with respect to the project schedules. They should be traced and reviewed in the next meeting; furthermore, corresponding discussions and resolutions may be generated to keep the project advanced.

Meetings contain enormous amount of information about the organizational culture, decisions, and information from previous meetings, etc. This information should be inherited by their successive meetings to maintain the organizational memory of the current project. For example, the resolutions reached in the previous meeting should be reviewed for their execution status; the information of related meetings could be easily accessed for reference, etc. In this way, a powerful conference minute system, which should provide automatic mechanisms to create the meeting minutes for recording important information generated from the meeting, to link relevant information among related meetings, and to provide a user-friendly interface for the access of the organizational memory, is greatly helpful to the project participants in increasing the understandings for the project and in turn, to accelerate the workflow of the project significantly. In other words, it is beneficial to the increase in productivity and the decrease in cost for their collaborations.

1.2. Previous works

There were three related researches [12, 16, 22, 31] on exploring the representations of the collaboration process and the mechanisms for recording the information in the meeting. First, the gIBIS system [12], which is based on the IBIS model [12], has been developed for use on large, complex design problems. This hypertext system used a relational database to store the group collaboration process as typed IBIS networks on a LAN. The IBIS network was structurized as cycles of three activities, i.e., ISSUE, POSITION and ARGUMENT, and several rhetorical moves to generalize, specialize, respond to, question, argue, and so on, among these activities. Users of this system must manually generate these activities and moves with text. Practically, this kind of network does not exactly match real behaviors and progresses of the meeting, which follows the agenda. It is inconvenient for the users because they must intentionally follow the model, which interrupts the actual cooperative work among them. Moreover, recording text only is not sufficient in current multimedia environment. Lots of applications, e.g., the video conference system and the web browser, have been or will be used during the conference. Multimedia data generated by these applications should be recorded to preserve ideas for future collaboration.

The second system [16] developed by the Toshiba Company is not suitable for this kind of environment, either. It was used with the audio conference tool. This system provided the participant a pen to take notes on the electronic display device of the pen-based portable computer. The note could be a keyword or non-character such as a mark. Notes were recorded and hyperlinked with the audio conversation at that moment. After the conference, a participant could select a note to selectively play back the portion of the audio conversation recorded at that time. However, only the audio data of the audio conference

tool was recorded. Other important data, such as the visual information, facial expressions or gestures of the participants in the video conference system, the shared objects and their corresponding operations on the shared applications, etc., were totally lost. Moreover, participants had to structurize the notes by themselves. The formal structure of the meeting progress was also lost such that users were unable to know it just by watching the hand-written note scripts.

Finally, the Jabber system [22] was primarily designed as a video archiving system to preserve progress of the whole video conference. The conference was recorded linearly and completely. A speech recognition module was built in this system to parse the audio data and created indices for them on real time. After the conference, a user could query the system to retrieve the corresponding meeting information, such as the audio/video data of the video conference and the text they exchanged at that time, using these indices. This system helped on finding the correct portion in the linearly recorded and enormous amount of video and audio data. However, the formal structured progress of the meeting, including important events in the agenda, is also absent. People can only recall some keywords from their memories to query for the information, which causes problems for the reviewers who did not participate in the meeting. Supports for multimedia applications are still insufficient.

In this paper, a multimedia World Wide Web based conference minute system is proposed and implemented to record and replay all formal and informal multimedia conference information with a well-structured approach, i.e., the structured progress flow, in web documents. It also maintains hyperlinks to related minutes to accelerate the progress of the project.

1.3. Paper organization

The paper is organized as follows. In section two, four design considerations for the conference minute system are discussed to build the Super Browser [40] platform in the multimedia environment, where the web browser, shared applications and video conference system are used for the meeting. Relationship with the World Wide Web (WWW) is achieved by the proposed HTML [46] conference minute format to preserve all important meeting information. In section three, the new proposed design consideration, i.e., the structured progress flow of the meeting, is introduced there. Issues such as how to create and maintain the flow as the meeting advances, object fields of nodes in the flow, etc., are discussed there. In section four, the four-layer architecture of the multimedia WWW-based minute system is proposed on the Super Browser platform. Recording and replaying operations are illustrated there. Implementation issues and current system status are presented in section five, with three real system snapshots accompanied. In section six, several important issues are discussed to improve this system. Lastly, we make the conclusion in section seven.

2. Design of the conference minute system

2.1. Four considerations of the minute system

Meetings usually convene with a preset agenda. Attendants follow the agenda to run the meeting. Issues are proposed. Many propositions, which are proposed with respect to these

issues, will create branches in the flow. They are raised to wait for decisions after deliberate debates. Many branches may be abandoned after discussions and then the flow may trace back to other branches. Conflicts may be resolved by the voting process. Repeating these events over and over, final resolutions may be reached. However, because the actual behavior of the meeting does not strictly conform to the meeting agenda [24], recording the agenda only is not enough. Actual progress flow should be recorded to reflect the actual content.

Information, which should be recorded in the conference minute of a multimedia meeting, can be classified into two classes, one is the *formal* class and the other is the *informal* class [16]. The formal information is those which represents specific events or objects in the meeting and can be written down with symbols or characters. Examples of the formal information are the date and time of the meeting, names of the attendants, the chairperson, the main motion and the resolutions of the meeting. Traditional conference minute usually writes down the formal information manually with text by a scribe. Nevertheless, not all formal information is included in traditional conference minutes. For example, the reasoning of a decision process, the rejected ideas, the alternatives of the propositions and some other valuable knowledge which are not part of the final decision but maybe helpful to understand how and why the decision is reached in its way are usually ignored in traditional minutes.

On the contrary, informal information which could not be easily described by handwritten symbols and characters, such as the process leading to a decision, atmosphere and nuances which reflect the behaviors of the participants, is lost in traditional minutes. Informal information usually embeds on the application contents that are affected by user inputs and the audio/video data generated from the multimedia applications, especially from the video conferencing system. Hence, a good conference minute system must provide mechanisms to automatically preserve and organize all formal and informal information in the conference minute such that the important information can be reused for subsequent work easily.

Three considerations for the conference minute system have been proposed in [50] as *completeness*, *ease of use and quick reference*. From experiences of the three previous systems, a good conference minute system must provide ease-of-use mechanisms which automatically record the multimedia meeting information as completely as the user needs and quickly refer to the interested portion of information. As discussed above, the informal information, which includes the audio/video data of the video conference and contents of the shared applications, should be preserved. The actual progress flow of the meeting, which is based on the conference agenda, should also be preserved to keep track of the actual conference logic, even with formal information which would be lost in traditional minutes such as proposition alternatives and debate branches not arranged in the agenda. Under this integration requirement to record both the progress flow and informal information, all nodes in the progress flow are hyperlinked with its associated informal data and further, are recorded in the conference minutes.

Due to experiences from the Jabber system, it is impractical to linearly record the entire conference not only due to the diverse types of recorded information and the needed enormous storage spaces, especially for the video frames, but also due to the difficulty in accessing any specific portion of the meeting with its poor-organized nature. Audio and video data, which reflects the atmosphere and nuances of the meeting, the operations on the shared applications, and the navigation on the WWW using the web browser should be

preserved only when necessary, i.e., at the time whenever an important event is raised and discussed in the meeting. All these events should be recorded in a well-structured form, i.e., the progress flow, as the meeting advances. As the gIBIS system asked for a method to strcuturize the collaboration process, one new important consideration for the conference minute system is how to build and maintain the *structured progress flow* of a meeting in the minute. With this kind of representation, all lost formal and informal information in the traditional minutes can be preserved completely. Details of issues for the structured progress flow will be discussed in Section 3. All these four considerations are included in our system design and implementation.

2.2. Platform of the conference minute system

Based on our previous system [7] on the CSCW platform [6], we are focused on the design of the conference minute system and the associated platform on which group project participants hold their meetings across the distance among them through networks. The system can preserve and archive important meeting information in the structured progress flow, and provide mechanisms to retrieve and review them from the project repositories or databases with an ease-of-use interface. Under the four considerations discussed above, we build our conference minute system on the more and more popular network environment, the Internet with the World Wide Web. This WWW environment provides the uniform interface and protocol for information searching on the WWW using the web browsers and the capability of integrating many diverse types of multimedia data generated from lots of applications used in the project meetings. These multimedia applications used in the meeting include video conferencing systems, web browsers, shared whiteboards and other shared applications. Our platform, i.e., the Super Browser [40] built on the Java application sharing framework [51] which supports the application sharing capabilities in the meeting, will be mentioned in the following.

2.2.1. The Super Browser platform and its Java application sharing framework. The primary goal of developing the Super Browser platform in our Lab. (Communications and Multimedia Lab. of National Taiwan University) is to build a platform which provides users a multimedia and network environment, through the WWW and Internet technologies, to enhance their collaboration activities. It is based on our proposed Java application sharing framework. Java enjoys the property of "write once, run everywhere" [41]. The combination of an application sharing system and Java allows us to "write once, run everywhere and cooperatively" [51]. There are two application sharing models [27], namely the event-sharing (input-sharing) model used in [24] and the request-sharing model in [6, 11, 26]. The request-sharing model starts an application on a server and multiplexes its graphical inputs to all the participants. The event-sharing model starts the same application with every participant, then broadcasts the input events for the applications. Current Java JDK (Java Development Toolkits), i.e., JDK 1.0 and 1.1, uses a peer model to delegate the look-and-feel of the AWT (Abstract Window Toolkit) components to a set of underlying "peer" classes. The request-sharing model cannot be implemented by intercepting graphical commands in pure Java such that the event-sharing model is adopted in the framework.

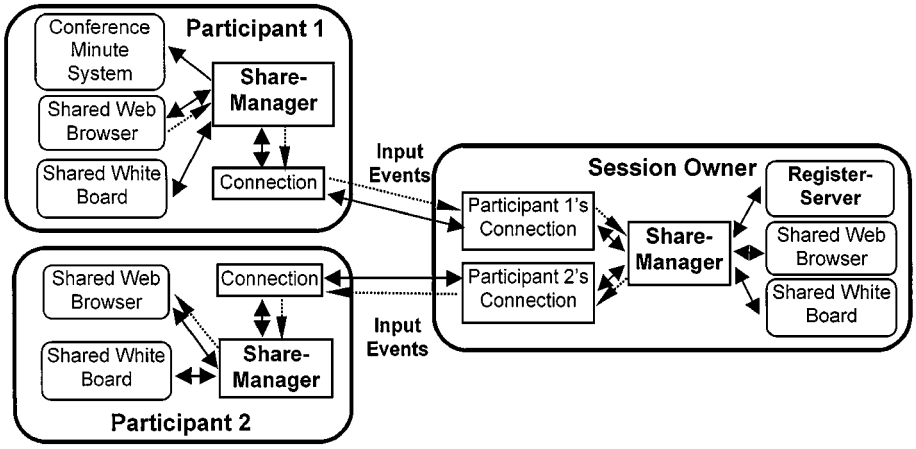


Figure 1. An on-going conference session with applications upon the Java application sharing framework. The dashed lines are the event passing flow.

An on-going conference session upon the framework is shown in figure 1. All shared applications in the session are written in Java. The participant who convenes the conference is the session owner. Other participants are session participants. The *RegisterServer* maintains the conference information centrally, such as the membership, participant IDs and the status of all ongoing activities. Whenever a participant wants to join the conference, he/she must register himself/herself with necessary information to the RegisterServer through the *ShareManager*. Only the application floor holder can generate input events. The session owner can revoke the application floor from a session participant who has been granted the floor. For checking privileged events such as the application floor request, generated events are first transmitted from the ShareManager of the floor holder to the ShareManager of the session owner through the *connection* module between them, then are broadcasted to all other participants. In this way, application contents of all participants can be the same. The scenario is shown in figure 1. In this figure, if the current floor holder of the shared web browser is participant 1, only he/she can control the browser with input events. These events are first sent to the session owner, then to participant 2 for generating the same browser content. This flow is shown as the dashed lines in figure 1.

During the duration of the collaboration activities, people can use the video conference system to see and hear each other, the shared whiteboard to express their ideas, the web browser to navigate on the WWW for accessing valuable information, and the shared applications to co-work on a task. The Super Browser platform, which is shown in figure 2, supports the shared web browser, shared whiteboard and other shared applications on the Java application sharing framework. Video conferencing capability is also provided in the platform. Because of the requirement of the event-sharing model in the Java application sharing framework, the participant at each site must run the Java same applications in the conference.

Two types of applications, which are classified depending on their communication behaviors with their peer applications, are provided on the platform. The first type is those which

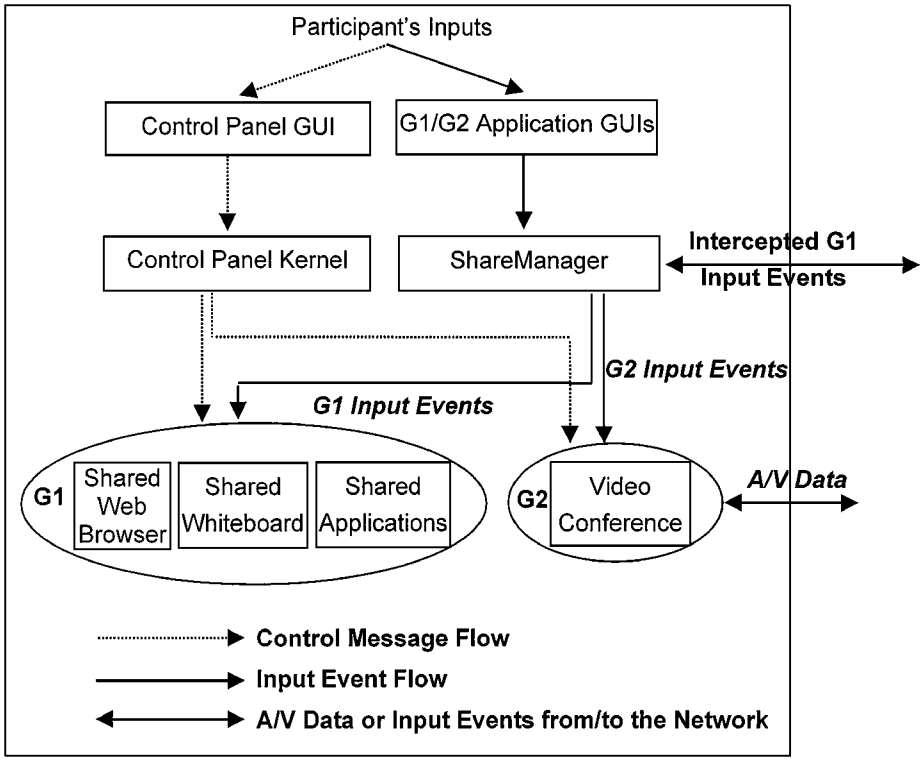


Figure 2. Platform architecture of the Super Browser platform and its message flows.

maintain the same application contents through the ShareManager of the session owner. These applications, such as the web browser, shared whiteboard and shared applications, are built on the Java application sharing framework and named as type G1. As mentioned above, the event-sharing approach achieves the What-You-See-Is-What-I-See views of all G1-type applications at each site. On the other hand, the video conference application (with video and audio data only), which creates the fully connected connections among all conference participants using the conference information stored at the RegisterServer, is named as the G2-type application. Without the need to transmit audio/video data first to the session owner, one instance of the video conference at each site handles the transmission/receiving process of the audio/video data to/from all other participants in the conference by itself.

Operations of the Super Browser platform are summarized in the following. When a conference is convened and joined by conference participants, they could execute applications supported in the conference such as the video conference, the conference minute system, etc., through the control panel user interface. The kernel of the control panel then invokes the applications to wait for the participant to input events for operation. Through the graphical user interfaces of the applications, users input events to use these applications. GUI events to G1-type applications are intercepted by the ShareManager built in the Super Browser platform. The ShareManager replicates these G1-type events, accompanying with

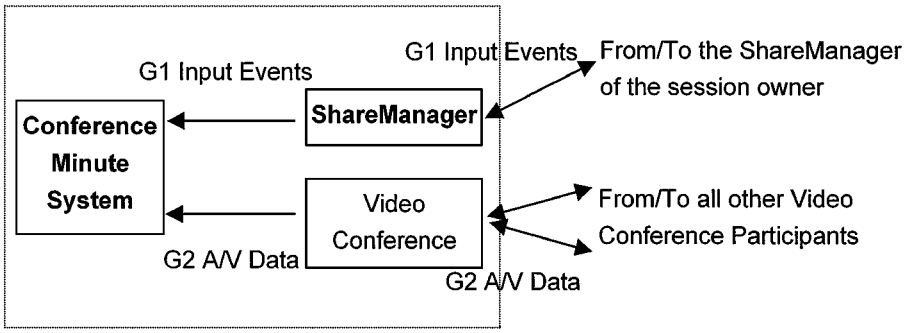


Figure 3. Relationship of the conference minute system with modules in the Super Browser platform.

different application identifiers, transmits one copy of them to the ShareManager of the session owner, and then broadcasts to all other participants as described above. At each site in the conference, these events are passed to the corresponding G1-type applications for execution according to the application identifiers of them. Flows of the control panel messages and application user events are also shown in figure 2.

Our conference minute system records both G1 and G2 types of information and other meeting information in the minute. For replaying the operations of the shared applications, our event-sharing approach records only input events and re-executes them for the replay [24]. Implementation details of the operations will be discussed in Section 5. However, for the replaying process of the request-sharing approach which needs to record a large amount of detailed window requests and replies generated from the underlying window system for each window operation, many critical data needs to be changed to conform to the current configuration of the window system. For example, in the X Windows environment, resource IDs, atom numbers, window coordinates, pixel values, key codes and sequence numbers are necessary to be mapped to new ones [14]. Comparing to the complex replaying process needed for the request-sharing approach, our event-sharing approach, which only records input events to applications, saves the needed storage space and recording/replaying overhead for the conference minutes significantly. The relationship among the conference minute system and modules in the Super Browser platform is shown in figure 3.

2.2.2. Design of the WWW conference minute. By providing an user-friendly interface, network and operating system transparency [4], and powerful search engines for accessing the enormous amount of information on the Internet, the World Wide Web technology has been the most successful one. Diverse types of media, e.g., hypertexts, images, VRML objects [43], downloadable Java programs [39] and, recently, audio and video [48], are integrated into the web contents for retrieval and viewing. Commercial Internet telephone and videophone products, such as VDOLive [42], Streamworks [50] and RealAudio [29], allow users to access real-time audio and video over the web. Netscape LiveMedia [28] framework further provides an open architecture to bring real-time audio and video to the web environment. Further, collaborations over the web [49], such as annotation and knowledge representation for information searching and sharing [4, 33, 34], have been set

as one of the primary directions in the future development process of the web. The web technology actually provides dispersed project participants an enabling platform to execute their collaborative projects efficiently and a repository to store project information on the Internet. Hence, the HTML minutes, which are automatically generated after the meetings by the conference minute system, can be more easily accessed and reviewed using the rapid-evolved multimedia WWW technologies.

In our HTML minutes, we integrate both the formal information, e.g., the meeting progress flow based on the agenda and the resolutions of the meeting, and the informal one, e.g., the multimedia data generated from the multimedia applications, into a single and complete conference minute which is stored in the WWW and can be accessed by group members all over the world using the web browser. When reviewing the conference minute, people can quickly understand the logic flow of the meeting by tracing the structured progress flow of the conference. With the hypermedia capability provided in the minutes, the reviewer can select any part in the progress flow he/she wants to see. In this way, what the participants said and behaved, what operations they took on the objects of shared applications, how the resolutions reached or arguments raised, and furthermore, the nuances in the meeting, can be replayed to reflect the actual situation at the recording time. More importantly, other related minutes and resolutions of the project can be easily accessed through the hyperlinks between two related meetings to concrete the project organizational memory.

Following the discussions about the information necessary to be kept in the conference minutes above, the format of the minute shown in figure 4 is composed of four fields, i.e., the *header*, the *progress flow*, the *resolutions* and the *list of related minutes*. The

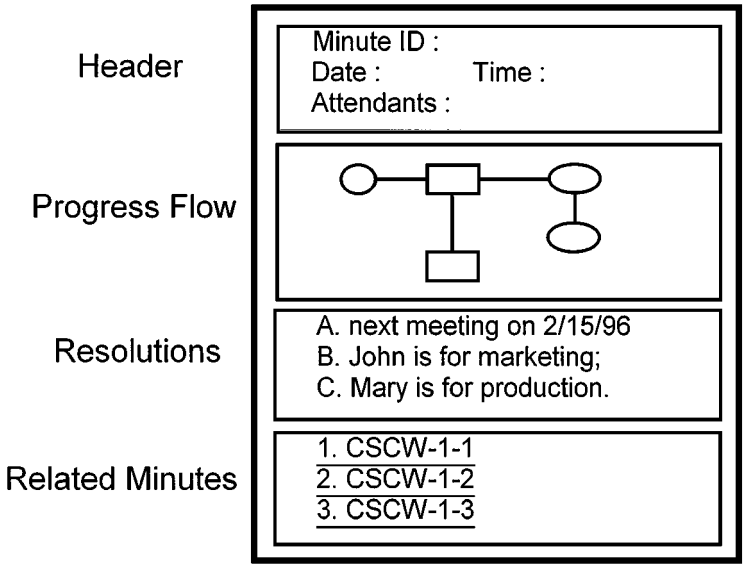


Figure 4. Format of the HTML conference minute.

conference minutes are stored in an HTML standard format which could be accessed by using any kind of web browsers across the Internet. The header contains information about this meeting, such as the time and date of the meeting, meeting attendants, the *MinuteID*, etc. The MinuteID is given by the conference minute system and is used to uniquely identify the minute of a specific meeting. The resolution field lists all resolutions reached in the meeting. Both the header and the resolution fields record the formal information. The progress flow field preserves the structured flow of the meeting and records the informal multimedia data during the collaboration. Operations performed in recording the flow of progress will be described in detail later in next section. At the end of the minute is the list of related minutes which provides hyperlinks to refer to other minutes.

Furthermore, important information of the current meeting minute must be inherited from the preceding one. For example, the date and time of current meeting are acquired from the preceding meeting's resolution for next meeting, the primary issue of current meeting is obtained from the resolutions field of the preceding one, the list of related minutes is also inherited from the list of the preceding one with the preceding one added, etc. When a new meeting is held, the minute system must generate a minute template for it which includes all necessary information about this on-going meeting.

In the following section, we focus on the design of the progress flow in the minute. We will first address the fundamental concept of the progress flow; then how the flow grows with the meeting and links with the associated multimedia data is described.

3. Design of the structured progress flow

In the representation of the progress flow, different events are shown as nodes with different shapes and the route taken by each proposition as branches. These events often begin with the keywords spoken by the session owner like *issue*, *proposition*, *assertion*, *argument*, *voting*, etc. Recording operations of the progress flow could be taken by a scribe. He/she maintains the flow of the progress and adds a new node at the correct location in the flow by clicking the mouse to choose the corresponding type of node in a keyword list whenever a keyword is identified. This is a purely manual approach.

In contrast, the natural language recognition techniques, as used in [22], can be applied to the keyword identification process to reduce the intrusiveness and overhead to the scribe significantly. The system automatically parses the spoken sentences of the participants to find the keywords. If a keyword is identified, a new node with corresponding type is created. In this case, the scribe only needs to link the newly created node at the correct location of the flow. Any of the participants could handle this easy minute-recording process, without the need of a professional scribe. However, this approach introduces great amount of computation efforts for language recognition.

When a keyword is identified, a new node of its type is automatically created and put on the correct location in the flow by the scribe to record the logic of the meeting. Each node is associated with its unique *NodeID* to represent its event type. Corresponding audio/video data and input events are recorded with associated timestamps to form a minute object which is hyperlinked with the node. Application *contents* and *status* when the recording process starts must be preserved with the successive input events such that the actions of the

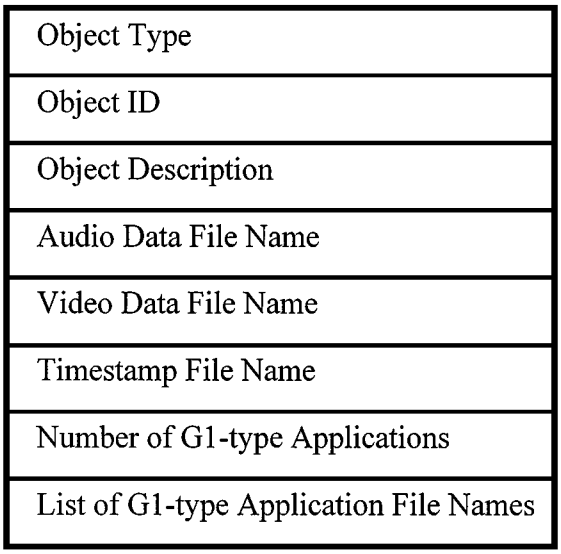


Figure 5. Data fields of the event object in the progress flow.

G1-type applications could be replayed by re-executing the recorded input events based on the application contents and status. The minute can be reviewed by clicking the mouse at the nodes to replay the G2 audio/video and G1 input events synchronously using timestamps. With this mechanism, people can understand what happened at the time of that event.

The format of the object is shown in figure 5. Each object linked to a node has its own unique ID (*ObjID*) to identify it internally in the minute system. It needs a mapping between the external NodeID and the internal ObjID when recording or replaying the media data of the object. The Object Type field is used to describe the type of the event whenever a keyword is identified. The Object Description field contains text to describe the event, such as an issue for electing the chairman of next meeting. Other data fields describe the recorded information. First two fields indicate the file names in which audio and video data are stored. They are named as "*ObjID*.audio" and "*ObjID*.video" respectively. Similarly, user input events to applications are stored in different files named as "*ObjID*.wb" for the web browser, "*ObjID*.sw" for the shared whiteboard, and "*ObjID*.ap1", "*ObjID*.ap2", "*ObjID*.ap3", etc., for the shared applications. Depending on the number of the actual G1-type applications used, the number of fields used for the application file name varies. Finally, timestamps for the recorded information are stored in the file named as "*ObjID*.ts" to achieve synchronization in replaying.

In the progress flow, different shapes of nodes represent different events which are identified by special keywords. The NodeID is shown beside the node to stand for its type as a01, b01, respectively as in figure 6. The root of the flow is located at the upper left corner. All nodes located to the right of a specific node and linked with lines are the children events which were happened after and related to the specific one. All nodes branched vertically below a specific node are its alternative sibling events, rooted from the same parent node.

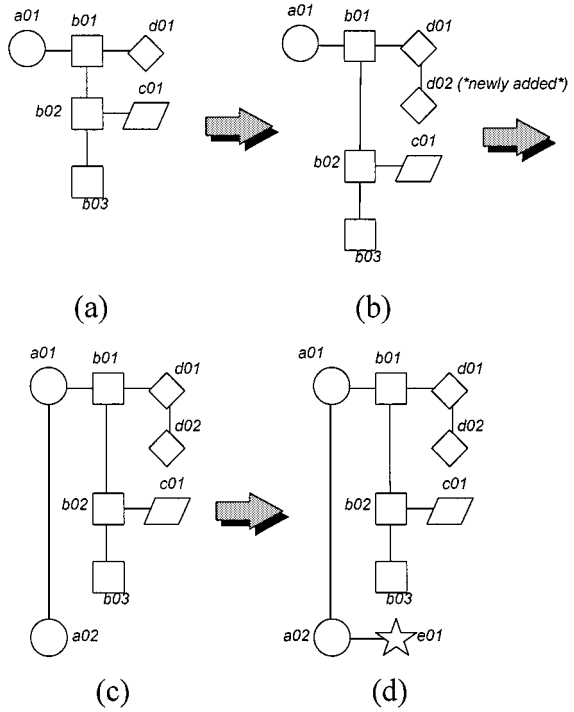


Figure 6. The growth of the structured progress flow.

As the flow grows, *flow collision prevention* algorithm must be performed to maintain the correct representation of the progress. Four stages of the flow are shown in figure 6. The initial flow is shown in (a). As node d02 is added into the flow, the system must be capable of performing the collision prevention algorithm to detect the collision between c01, the old node, and d02, the new one. Then the flow rearranges its layout to accommodate the new node. The result is shown in (b). The new flow after node a02 is added without rearranging the layout as shown in (c). In (d), it does not need any extra processing to add node e01 into the flow.

Consider the scenario of figure 6. The circle nodes, i.e., node ID with prefix "a", are issues; the rectangle nodes, with prefix "b", are the propositions for their parent issues; the parallelogram node, with prefix "c", stands for the withdrawal from its parent proposition; the diamond nodes, with prefix "d", are debates with respect to their parent propositions; and finally, the star node, with prefix "e", represents the amendment to the issue. The following shows an example for the progress flow of a meeting. In (a), issue a01 (destination of the Sunday picnic) is set. Three propositions, b01 (the City Park), b02 (the Sun-Moon Lake) and b03 (the Big Forest), are proposed. The proposition b01 is discussed at node d01; the proposition b02 is withdrawn at node c01; and no further actions are done on proposition b03. In (b), a new debate, i.e., node d02, is added to the proposition b01 about extra reasons to support the choice of the City Park. In (c), the chairperson suspends the

issue a01 because he/she thinks the debates has spent too much time on the choice for the place. Then he/she leads the meeting to a new issue a02 (when to start the new project). In (d), a participant asks for an amendment for this issue to discuss the purpose of the new project and its relationship to its previous project.

Let us summarize advantages of this structured progress flow design. First, it preserves both the formal and informal data in the flow. Important ideas and other information are not lost, which satisfies the completeness requirement. Second, this system is easy to use because the work people should do is only to put the newly generated event nodes on the flow. All other works are handled automatically by the system. Third, people can refer to any event in the progress flow quickly just by clicking the node representing the event. The event object is accessed through the hyperlink of the node. Fourth, the advance of the meeting can be well-controlled by the graphical representation of the flow, i.e., the structured progress flow. Finally, the minutes can be accessed from the WWW by the absentees or other related people of the project all over the world to accelerate the execution process of the project.

4. System architecture of the conference minute system

System architecture of the conference minute system is shown in figure 7. It is a four-layer architecture which is composed of the graphical user interface, the progress flow/ description/template managers, the minute object manager and the media recorder and player. The graphical user interface displays the minute and provides users functionalities to maintain the minute contents. The minute object manager maintains all existing conference minutes and creates a new minute with its MinuteID in response to the requests of the template manager. Whenever a meeting is established for group works, the template manager gets the content of the last meeting by the MinuteID of the last meeting and generates an appropriate template for the minute of this meeting through the primitives provided by the minute object manager. If this meeting is the first one of the project, only the header field of the minute is filled. Otherwise, besides the header field, the template manager further parses the contents of the minute of the last meeting to generate a minute template which

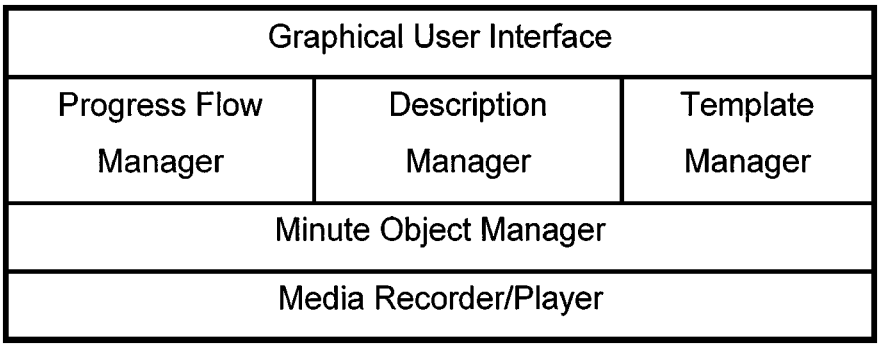


Figure 7. The four-layer architecture of the conference minute system.

Table 1. Primitives provided to the graphical user interface by the template manager.

Primitive name	Input parameter	Return value	Description
New_Minute	The last MinuteID (optional)	This MinuteID	Create a minute when a new meeting is held
End_Minute	MinuteID	Success/Error	Close the minute when the meeting ends

Table 2. Primitives provided to the template manager by the minute object manager.

Primitive name	Input parameter	Return value	Description
New_Template	The last MinuteID	This MinuteID	Create a new minute
End_Template	MinuteID	Success/error	Close the minute

inherits the links to the related minutes and all uncompleted resolutions from the minute of the last meeting, as described above. These primitives are listed in Tables 1 and 2.

During the meeting, the description manager or the progress flow manager are executed depending on the operations requested in the user interface. The description manager is responsible for handling descriptions of all nodes in the flow. Brief descriptions can be added to the objects on-line if the scribe does not have enough time to write down the complete descriptions; they can be modified after the node is created. The progress flow manager assigns new NodeIDs to the newly created nodes and performs the flow collision prevention algorithm to maintain the progress flow shown in the user interface and further, provides hypermedia capability to the associated object managed by the underlying minute object manager. The minute object manager maintains a mapping table between each pair of the NodeID and the ObjID. It first converts the NodeID of the node in the progress flow or the description manager to the ObjID for accessing the object.

Depending on the types of the operations, i.e., the recording or replaying process, the minute object manager takes appropriate actions to handle the object. In the recording process, the media recorder connects to the ShareManager and the video conference system to intercept input events for each G1-type applications and the audio/video data, respectively. According to their temporal relations, it generates timestamps of the recorded information. These data are stored in files by the media recorder; thus the object manager could complete the object fields with corresponding file names. Similarly, in the replaying process, the minute object manager retrieves the information file names and its associated timestamps from the object. With the help of timestamps, the media player can access appropriate portions of the recorded information from the media files and send them to the G1-type applications for generating the same application contents or to the video/audio player application for replaying the recorded audio/video data synchronously.

When a keyword is identified automatically using the language recognition technique or chosen manually from the keyword list, a new node and its associated object are created by calling the primitives in Tables 3 and 4. It then triggers the information recording process. Audio/video data from the video conference and the input events from the G1-type applications are recorded with timestamps by the media recorder using the primitives in Table 5. All the data could be stored in files and be accessed by using the corresponding

Table 3. Primitives provided to the graphical user interface by the progress flow manager.

Primitive name	Input parameter	Return value	Description
Add_Node	Node type	NodeID	Add a new node in the progress flow
Choose_Node	NodeID	None	Choose a node in the flow to replay

Table 4. Primitives provided to the progress flow manager by the minute object manager.

Primitive name	Input parameter	Return value	Description
Add_Object	NodeID, Node type	ObjID	Add the corresponding object of the node
Choose_Object	NodeID	None	Choose the node to replay

Table 5. Primitives provided to the minute object manager by the media recorder.

Primitive name	Input parameter	Return value	Description
Media_Record	ObjID	Audio File Name, Video File Name, TimeStamp File Name, Number of G1-type applications, List of G1-type application File Names	Record the audio/video data, user input events of G1-type applications and timestamps to corresponding files

Table 6. Primitives provided to the graphical user interface by the description manager.

Primitive name	Input parameter	Return value	Description
Edit_Node_Description	NodeID	None	Edit node description

Table 7. Primitives provided to the description manager by the minute object manager.

Primitive name	Input parameter	Return value	Description
Store_Obj_Description	NodeID, *Description String*	Success/Error	Store the description into object field

file names in the fields of the object. The description of the node can be added or modified with the primitives provided by the description and object managers, as shown in Tables 6 and 7, respectively. The object manager collects information such as the descriptions from the description manager, the object type and ObjID, the progress flow from the progress flow manager, and the resolutions fields into the minute to complete the minute of this meeting. Finally, it is further responsible for converting the minute into the HTML format to be accessed through the WWW, as discussed before. The operations performed for the recording process when a new node is created are shown in figure 8.

After the meeting, the minute can be reviewed by group members and people having access rights to it using any available web browser. The reviewer can move the mouse to

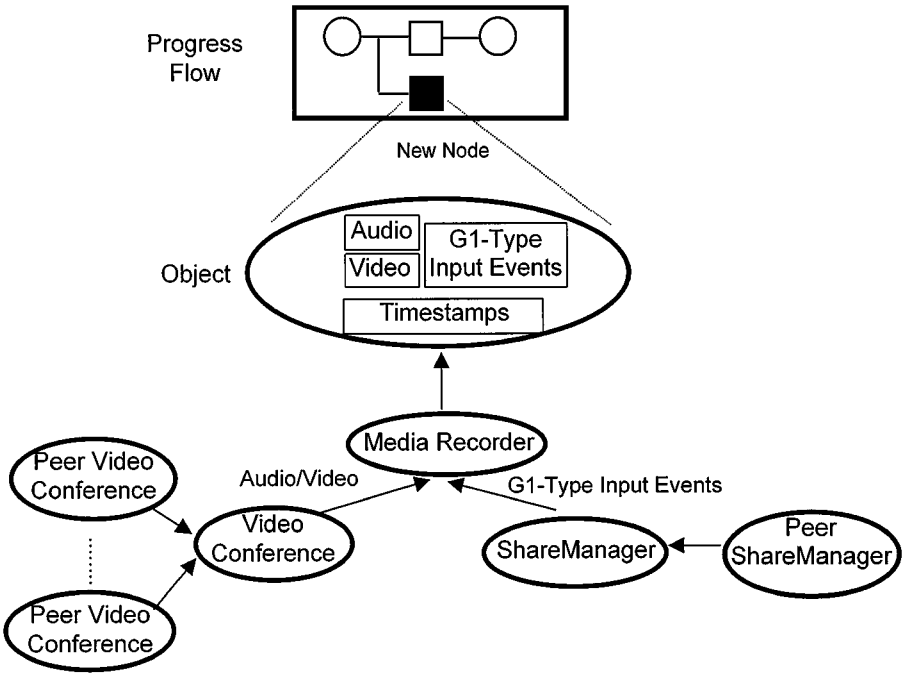


Figure 8. Operations of recording object data when a new node is created.

choose any node in the progress flow, with the primitives in Tables 3 and 4, and click the mouse to replay the data contained in the associated object. As soon as the object manager is notified about the replaying operations, it may execute an audio/video player for replaying the A/V data and all needed G1-type applications according to the application identifiers of the input events. The object manager retrieves the data from the files indicated in the data fields. Then these data are handled by the media player to retain their timing relationships such that the replaying operations executed by the audio/video player and the G1-type applications can be synchronized. The *Media_Play* primitive provided by the media player and called by the object manager is shown in Table 8. When the replaying process of the recorded data in a node is completed, all applications executed for the replaying process are closed. The operation for replaying the object data is shown in figure 9.

Table 8. Primitives provided to the minute object manager by the media player.

Primitive name	Input parameter	Return value	Description
Media_Play	Audio File Name, Video File Name, Timestamp File Name, Number of G1-type applications, List of G1-type application File Names	Success/Error	Replay the audio/video data, and user inputs to their associated G1-type applications synchronously using timestamps

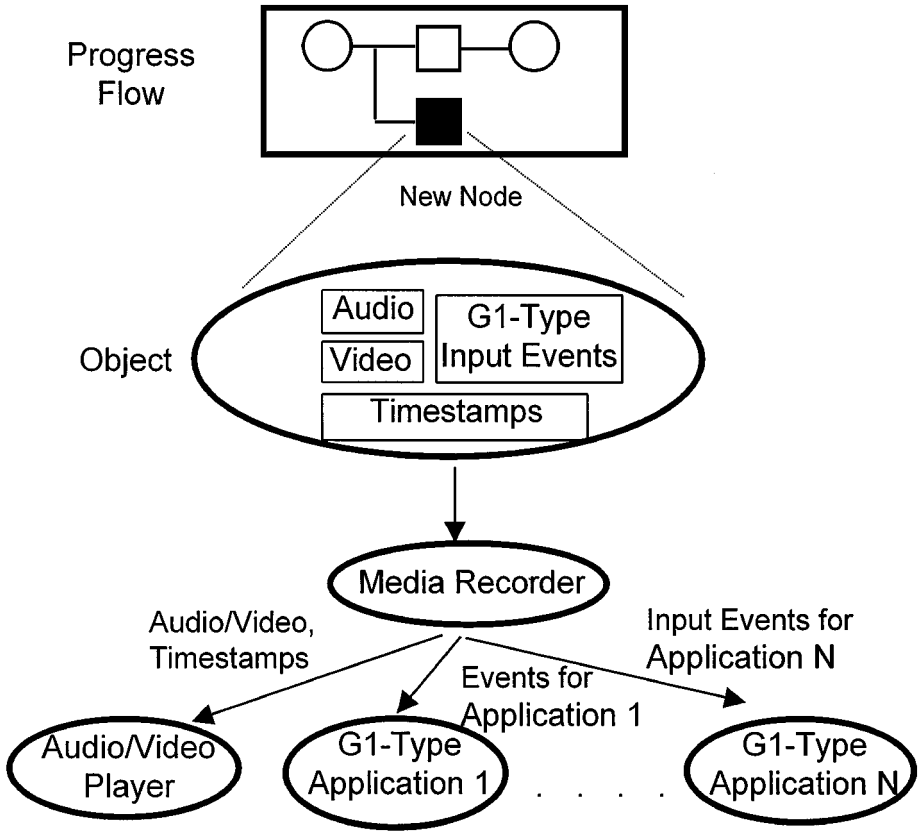


Figure 9. Operations of replaying object data when a node is clicked.

5. Implementation issues and current status

5.1. Implementation issues

The conference minute system is built on the Super Browser platform, which uses Java JDK 1.0 in the Java application sharing framework. This platform is implemented on workstations connected through an FDDI and an Ethernet network. The operating system used is Sun Solaris 2.5 and both X Windows and Openlook are chosen to build the user interface. For facilitating the video conferencing capability on LAN, each workstation is equipped with a camcorder, a microphone, a speaker and a video compression/decompression board, which provides motion JPEG compression and decompression functions for both image and video data.

Current implementation for the media recorder of the conference minute system is executed by two UNIX processes. One is the Java-written ShareManager for intercepting input events of Java applications, such as the already implemented shared web browser and the

shared whiteboard. The other is the C-written AV recorder for the audio/video data of the C-written video conference. All the conference participants can use the shared whiteboard simultaneously. However, only one participant can have the floor of the shared browser at a time. A simple floor control mechanism is provided in current implementation. The session owner has the privilege to revoke the floor at any time. Initially, the session owner has the floor. Other participants can request this floor from the session owner. The session owner decides whether to release the floor or not. If requests from different participants come at the same time, the session owner can choose any participant to get the floor, if the session owner grants the requests to release it. Whenever another participant needs the floor of the application, he/she must send a request to the session owner and wait for the session owner to revoke the floor from the original floor holder. This is similar to the way students used in the school class to ask the teacher for a chance to talk. For further discussions with the floor control policies and mechanisms, please refer to [37].

The ShareManager on each site is implemented by modifying the Java's Abstract Window Toolkit (AWT) class library [21] to capture the input events from the Java-written GUIs of the G1-type applications. Originally, a Java JDK 1.0 program must inherit GUI components and override the *handleEvent()* method to handle events. When an event happens. The *postEvent()* method of the target component propagates sequentially up the GUI hierarchy to call the handleEvent() method on the propagation path, until either one handleEvent() returns "*true*" to consume the event or the root of the hierarchy is reached. Hence, the postEvent() method is modified to intercept input events. The modified postEvent() returns "*false*" immediately if it is not the floor owner; otherwise it behaves as usual. If any component of the application floor holder consumes the event, the ShareManager transmits the event to the session owner, then broadcasts to all session participants, through connections between them. We view the GUI hierarchy as a tree, and encode the component as the path from the root window to the target component. When the ShareManager of the participant who is not the application floor holder receives an event, it tries to post the event until the event is consumed. In this way, each participant can have the same view for the shared applications. The encoded codewords of all input events are recorded with their timestamps in a Java log file by the ShareManager of the session owner [51]. These input events are classified into four types, i.e., the "Resize" type for the window resizing operation, the "Mouse" type for the mouse movement operations, the "Action" type for the URL operations, e.g., the "Forward" operation, and the "Event" type for other operations. Each recorded input event includes its type, timestamp, codeword of its Java window component, and other related information. One example of the recorded input event is shown in figure 10.

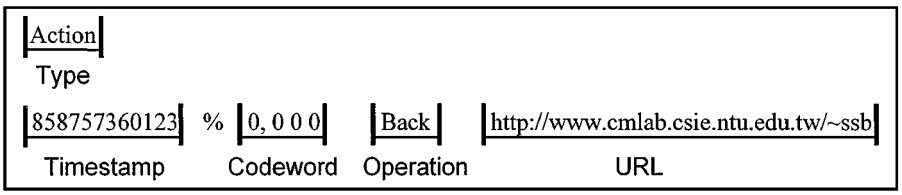


Figure 10. Format of the recorded user input events.

The other process, i.e., the AV recorder, is responsible for recording the audio/video data of the video conference. To avoid the interference with the normal audio/video transmission and receiving operations of the video conference application, this AV recorder performs the time-consuming disk I/O operations for writing the enormous amount of the recorded A/V data. Several pieces of shared memories are created for the video conference application to write out the A/V data whenever they are generated. The AV recorder periodically checks these pieces of shared memories to see whether there are A/V data. If yes, this AV recorder writes these A/V data and relative timing information to disk files for later replaying operations.

Whenever the conference ends, the conference minute is converted into its HTML format as described before. A Java Applet for the replaying operation is started when the HTML minute page is opened by the web browser. This Applet contains an image button. As the button is clicked, the Applet invokes the "*native*" C-written conference minute system to show the progress flow and other information. Reviewers can click any node in the flow to replay the recorded Java input events and A/V data synchronously. Two independent processes are needed for the replaying operation of the media player. One process, i.e., the Java player, invokes the same Java-written applications at the recording time to execute each user input events using its timestamp. Audio/video data is played back by the multi-user AV player.

Maintaining synchronization of the replaying process must let people know the temporal relationship of application operations and the logic flow of the meeting. In our current implementation, the Java log files and the A/V data files are stored in the same file system, which makes synchronization among multiple independent processes, i.e., the AV player and the Java player, can be achieved more easily. However, this restriction may not be practical for use. We will discuss this issue in next section.

There are several important issues in the current system should be addressed. First, recorded timestamps are used for achieving synchronization between the AV and Java players,. Two timing information, i.e., the start time and the finish time, is stored with each node in the meeting progress flow. For regenerating the same application content as it was at the start time of a node, the recorded Java events must be re-executed from the first one to the event with its timestamp just before the node start time. All the events with their timestamp values smaller than the start time of the node are executed without any delay between two events to create the application contents as fast as possible. Then, these two players should be executed synchronously during the replaying period, i.e., between the start time and the finish time of the node. Using the timestamps, the Java player executes the events according to the temporal relationship at the recording time. The AV player displays the video frames of the same participant on the same window with the normally recorded rate. Second, because the time for the replaying operations of video data and Java input events may not be equal to the time needed at the recording time and the operations for these two kinds of data consume more CPU time than the operations for the audio data, the replaying process for the video data and Java events needs further adjustment to keep synchronization with the audio data. Another reason is the non-real-time scheduling for multiple independent replaying processes in the UNIX operating system. It is not guaranteed that the replaying processes can be executed at the time needed for synchronization. From real experiences on the replaying operations, the result of media synchronization is not so precise but acceptable, if the system load is not high.

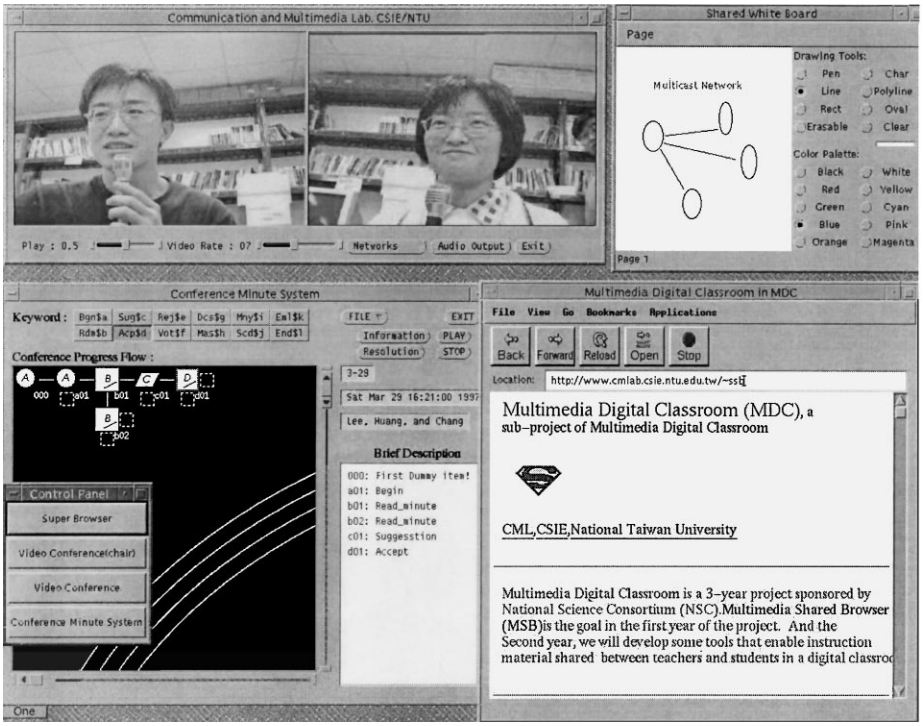


Figure 11. The conference minute system with the video conference, shared whiteboard and the shared browser at the recording time.

5.2. Current system status

The implemented conference minute system is intensively tested for the real meetings of our research projects. The snapshot of the conference minute system when recording is shown in figure 11, accompanying with the control panel, the video conference system, the shared whiteboard and the shared web browser. As discussed above, conference participants activate the necessary applications for sharing from the control panel, which is shown at the lower left corner in the figure. The video conference system, shown at the upper left corner, provides several video windows to display participants' videos with an automatically adjustable mechanism for the video frame rate according to the network traffic. The shared web browser, shown at the lower right corner, supports functions for the HTML 3.0 tag, VRML 1.0, MPEG audio/video and several kinds of image formats.

The conference minute system is shown at the lower left part in figure 11. Current implementation for the minute system only provides the scribe the keyword list to choose the keywords. Upper part of the minute window is the keyword list in which important keywords are listed. These keywords [32], for example, Bgn$a stands for the keyword *begin* and the ID type *a*, could be chosen manually by the scribe to add a node. It represents the

event which starts the conference with a primary issue. Left portion of the minute window shows the current progress flow of the conference and the right one is the brief descriptions of all nodes in the progress flow. Each node in the progress flow is shown with its NodeID and a different shape according to its keyword type. When a new keyword is identified, its node could be linked to an existing one by choosing one of the two dash-lined boxes to represent its relationship, i.e., the child or sibling node, with the linked node.

As the conference ends, a final HTML conference minute is automatically generated by the conference minute system. Four fields, i.e., header, progress flow, resolutions, and related minutes, are presented in the minute when opened by the shared web browser. A Java Applet, which is shown as a button around the middle part in figure 12, can be clicked to execute the conference minute system. Progress flow of the meeting is automatically loaded for reviewing. Thus, the reviewer can choose any node to play back the recorded information. The snapshot when a reviewer reads the conference minute with the

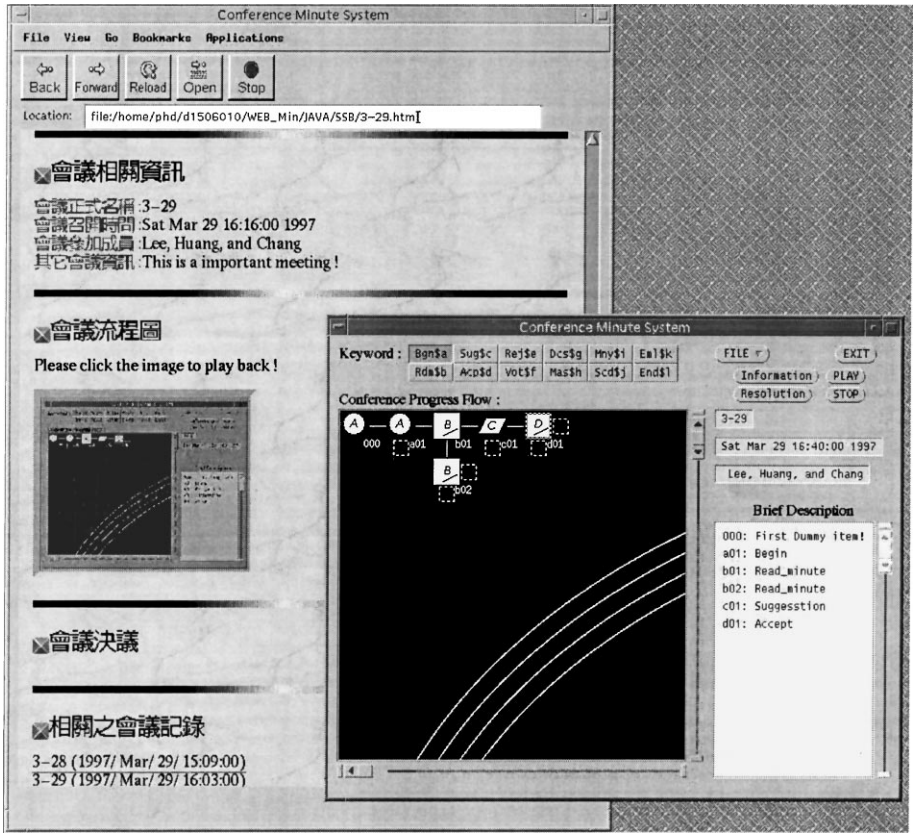


Figure 12. The automatically generated conference minute with the invoked conference minute system after clicking the Java Applet button.

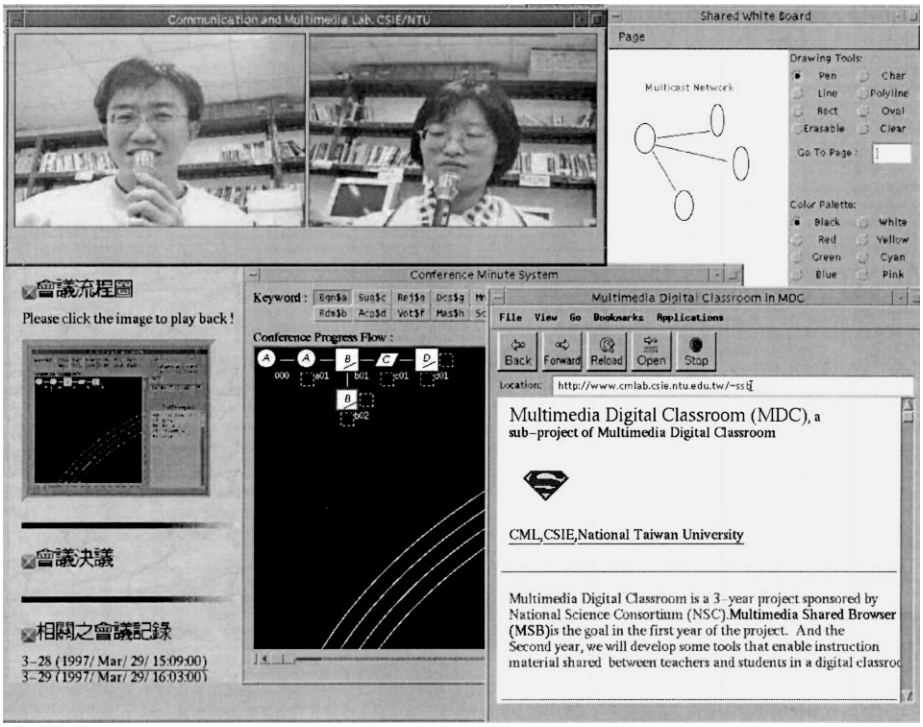


Figure 13. The replaying process of a node in the progress flow of the conference minute. The shared browser, shared whiteboard and AV player are replaying the recorded data.

shared browser and performs the replaying operation is shown in figure 13. The recorded audio/video data is replayed in the windows of the AV player, which is shown at the upper left part of the figure. Java events are executed by the shared browser and shared whiteboard, which are shown at the lower right and upper right corners in the figure, to play back the original operations at the recording time.

Under physical system limitation for numbers of shared memory pieces and the size of each piece, the video frame number which can be recorded per second is reversely proportional to the interval for checking the shared memory of the AV recorder. It means the longer the interval is, the less the recorded frame rate is. Their relationships are shown in Eqs. (1) and (2). From these two equations, it is possible to increase the recorded frame rate by shortening the checking interval, if other factors remain the same. Unfortunately, the shorter the checking interval is, the more CPU time the operating system spends on executing disk writing operations. Thus, the video conference system acquires less CPU time to execute its operations, which will degrade its audio/video quality. This system behavior causes a tradeoff between performances of the video conference and the AV recorder. Under this tradeoff, some accumulated video frames, which exceed the system limitation in the checking interval, are dropped. However, because only a small percentage,

e.g., 2 or 3%, of audio data loss is acceptable, audio data should be recorded completely.

$$\text{Size of the shared memory} = \text{size of each video frame} * \text{recorded frame rate} \quad (1)$$

$$\text{Recorded frame rate} = \text{recorded frame number per checking interval/the checking interval} \quad (2)$$

6. Discussions

In the following, we discuss some important issues to enhance the system performance and user acceptance of the conference minute system.

- **Disk storage:** Let us consider the amount of audio data if recorded completely in a one-hour meeting. With the 8 kHz audio PCM sampling rate, the audio data in one hour is about 3.6 MB, which is much larger than the capacity of a floppy disk. The amount of video data is much more than that of audio. Disk spaces will be exhausted in a very short time without applying any compression technique. DPCM, ADPCM or CELP algorithms could be used to reduce the audio data. Silence detection technique would also help. For video data, H.261, H.263, JPEG or MPEG video compression algorithms can greatly reduce the data amount up to one hundredth of the original one. The H.261 and H.263 standards are especially designed for the video conference system. Other algorithms, such as the scene change detection, model-based coding [2, 44], etc., are further beneficial to reduce the video data of the video conference. In current system implementation, the motion JPEG compression/decompression method is adopted to reduce the recorded video size. Further, comparing to the large amount of data by recording the window messages of the applications, our event-sharing approach which only records the input events saves the storage spaces significantly.
- **Streaming multimedia on the WWW:** As mentioned in previous section, playback operations through the WWW is much more difficult than what we have done in the same file system. The web protocol must support transmission of the real-time multimedia data, provide synchronization mechanisms for different types of media data, which may come from different sources, and so on. However, current web technologies cause problems when supporting the transmission of the real-time audio/video data [5, 9, 48]. For example, current Hypertext Transport Protocol (HTTP) protocol [47], which is based on the TCP protocol, used on the WWW is unsuitable to transmit the real-time constrained A/V data because it results in a variable response time. Hence, real-time protocols like the Real-Time Transport Protocol (RTP) [36] should be included in the minute system for the replay of the A/V data using the web browsers, as well as in the Internet conferencing environment with the video conference. The Real Time Streaming Protocol (RTSP) proposal [35], developed by the Internet Engineering Task Force (IETF) MMUSIC working group, provides a complete set of functionality for client-server streaming of multimedia, supports live or stored real-time audio/video data, and other specific benefits. When it is accepted as an Internet standard, the RTSP will provide a solution for streaming multimedia over the Internet.

- **More supports for organizational memory:** Besides the hypermedia support for the related minutes, the reached resolutions should be reviewed in the successive meetings to further link all information about the execution, status and results for the resolution. The resolution list could be inherited by the next minute as the way done for the list of the related minutes. Forward and backward links of the resolutions provide quick accesses to the reviewer. Moreover, new links could be added between the node where a certain resolution is reached or traced in the progress flow and its corresponding resolution. In this way, by tracing the links and then replaying the information recorded in the associated node for a resolution, our design builds a more concrete organizational memory for group collaboration.
- **International standards conformation:** Several international standards for building the conferencing services are proposed in recent years, such as the International Telecommunication Union (ITU) H.320 [17], H.323 [18], H.324 [19] and T.120 [20] series. These standards address the issues for building the multipoint multimedia conference environment, the multiplexing/demultiplexing functionalities, the interface specifications with the underlying networks, the control procedures executed, etc. In the near future, we will consider to realize our minute system on these conferencing environments, especially on MBone [25] to support real-time communication over wide area IP networks for the video conference system.

7. Conclusions

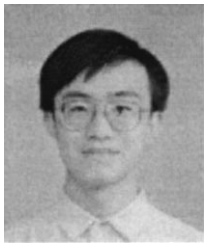
As people get more and more closely related with the advance of technologies, large projects with many geographically dispersed experts involved will become the normal working behavior of daily life. In this paper, we proposed a conference minute system which satisfies four important considerations and is based on the most prosperous Internet and World Wide Web environment. All the formal and informal information in the project meetings is preserved in the conference minute together with other information needed to maintain the project organizational memory. Viewing the progress flow, people can quickly understand the logic flow of the meeting and easily select whichever part he/she wants to see via the hyperlink between the node and its associated object. With the help of this conference minute system, people have the most complete information for the project such as its current execution status, the resolutions reached in any meeting, and any valuable information generated in the meeting to aid the advance of the project. In this way, the project execution and management could be more easily achieved to increase the productivity of the collaborative groups.

References

1. H. Abdel-Wahab and M. Feit, "XTV: A frame-work for sharing Xwindow clients in remote synchronous collaboration," in Proc. of IEEE Tricomm: Communications for Distributed Applications & Systems, Chapel Hill, April 1991, pp. 159–167.
2. K. Aizawa, H. Harashima, and T. Saito, "Model-based analysis-synthesis image coding (MBASIC) system for a person's face," Signal Processing: Image Communication, Vol. 1, pp. 139–152, 1989.
3. J. Baldeschwieler, T. Gute Kunst, and B. Plattner, "A Survey of X protocol multiplexors," ACM SIGCOMM Computer Communication Review, Vol. 23, No. 2, pp. 16–24, April 1993.

4. R. Bentley, T. Horstmann, K. Sikkel, and J. Trevor, “Supporting collaborative information sharing with the world wide web: The BSCW shared workspace system,” in Proc. of the 4th International WWW Conference, Boston, MA. USA, Dec. 1995, pp. 663–673.
5. J. Bolot and P. Hoschka, “Sound and video on the web,” Tutorial on the 5th International World Wide Web Conference, Paris, May 1996.
6. I.C. Chang, J.H. Huang, and W.H. Tseng, “Design and implementation of a multimedia CSCW platform,” International Journal of Multimedia Tools and Applications, Kluwer Academic Publishers: Norwell, MA, USA, Vol. 2, No. 2, pp. 85–109, March 1996.
7. Ing-Chau Chang, Bo-Shen Liou, and Jau-Hsiung Huang, “A hypermedia conference recording and processing system for group collaboration,” in Proc. of the 10th International Conference on Information Networking (ICOIN-10), Kyung- Ju, Korea, Jan. 1996, pp. 221–230.
8. T.M. Chen, C.C. Yang, W.H. Tseng, I.C. Chang, J.H. Huang, C.C. Lin, M.S. Lee, N.J. Fon, and S. Kaw, “Design and implementation of a desktop computer supported cooperative work system,” IEEE Trans. on Consumer Electronics, Vol. 40, No. 4, pp. 827–835, Nov. 1994.
9. Z. Chen, S. Tan, R.C. Campbell, and Y. Li, “Real time video and audio in the world wide web,” in Proc. of the 4th International World Wide Web Conference, Boston, MA, USA, Dec. 1995, pp. 333–348.
10. G. Chung and K. Jeffay, “Accommodating latecomers in shared window systems,” IEEE Computer Mag., pp. 72–74, 1993.
11. C. Chung, “The Recorder and Player of Application Execution,” Master Thesis, Dept. of Computer Science & Information Engineering, National Taiwan University, Taiwan, June 1996.
12. J. Conklin and M.L. Begeman, “gIBIS : A hypertext tool for exploratory policy discussion,” ACM Transcations on Information System, No. 6, No. 4, pp. 303–331, 1988.
13. C.A. Ellis, S.J. Gibbs, and G.L. Rein, “Groupware: Some issues and experiences,” CACM, Vol. 34, No. 1, pp. 39–58, Jan. 1991.
14. T. Gutekunst et. al., “A distributed and policy-free general-purpose shared window system,” IEEE Trans. on Networking, Vol. 3, No. 1, pp. 51–62, 1995.
15. J. Huang, C. Yang, W. Tseng, C. Lee, B. Tsaur, L. Chung, and W. Liu, “Design and implementation of multimedia conference system on broadcast networks,” in Proc. of 18th Annual Local Computer Network Conference, Minneapolis, 1993, pp. 337–341,
16. T. Imai, K. Yamaguchi, and T. Muranaga, “Hypermedia conversation recording to preserve informal artifacts in realtime collaboration,” in Proc. of ACM Multimedia, 1994, pp. 417–424.
17. ITU-T Recommendation H.320, Narrow-band ISDN Visual Telephone Systems and Terminal Equipment, 1993.
18. ITU-T Recommendation H.323, Visual Telephone Systems and Equipment for Local Area Networks which Provide a Non-guaranteed Quality of Service, Draft, Nov. 1995.
19. ITU-T Recommendation H.324, Terminal for Low Bitrate Multimedia Communication, Draft, Nov. 1995.
20. ITU-T Recommendation T.120, Data Protocols for Multimedia Conferencing, Draft , Oct. 1995.
21. JAVA Abstract Window ToolKit (AWT), <URL:http://ugweb.cs.ualberta.ca/nelson/java/AWT.Introduction.html>, 1996.
22. R. Kazman, R. Al-Halimi, W. Hunt, and M. Mantei, “Four paradigms for indexing video conferences,” IEEE Multimedia Mag., pp. 63–73, Spring, 1996.
23. C. Lee, “The design and implementation of virtual-X system in CSCW,” Master Thesis, Department of Computer Science & Information Engineering, National Taiwan University, Taiwan, June 1994.
24. R. Manohar and A. Prakash, “Replay by re-execution: A paradigm for asynchronous collaboration via record and replay of interactive multimedia sessions,” ACM SIGOIS Bulletin, Vol. 15, No. 2, pp. 32–34, 1994.
25. MBone (or IP Multicast) Information Web, <URL:http://www.mbone.com>, 1997.
26. W. Minenko, “An advanced application sharing system for synchronous collaboration in heterogeneous environments,” ACM SIGOIS Bulletin, Vol. 15, No. 2, pp. 40–44, 1994.
27. W. Minenko, “The application sharing technology,” The X Advisor, Vol. 1, No. 1, June 1995.
28. Netscape Communications Corporation. Netscape Announces New Real-Time audio and Video Framework for Internet Applications, <URL:http://www.netscape.com/newsref/pr/newsrelease81.html>, 1996.
29. Progressive Networks, Progressive Networks Announces Open RealAudio Architectures, <URL:http://www.realaudio.com/ prognet/openarch/announcement.html>, 1996.

30. M. Ramage, “Engineering a Smooth Flow? A Study of Workflow and Its Connections with Business Process Reengineering,” Dissertation for the degree of Msc in Human-Centred Computer Systems, School of Cognitive and Computing Sciences (COGS), Sussex University, Brighton, BN1 9QH, England.
31. G.L. Rein and C.A. Ellis, “RIBIS: A real-time group hypertext system,” International Journal of Man-Machine Studies, Vol. 34, No. 3, pp. 349–367, 1991.
32. H.M. Robert III and W.J. Evans, “Robert’s Rules of Order Newly Revised,” The Scott, Foresman Company, Chap. III, 1990 Edition, pp. 57–81, USA.
33. M. Roscheisen, C. Mogensen, and T. Winograd, “Beyond browsing: Shared comments, SOAPs, trials, and on-line communities,” in Proc. of the 3th International WWW Conference, April 1995, pp. 739–749.
34. M. Roscheisen, T. Winograd, and A. Paepcke, “Content ratings and other third-party value-added information defining an enabling platform,” D-Lib Magazine, Aug. 1995.
35. RTSP Resource Center, <URL:http://www1.realaudio.com/prognet/rt>, 1997.
36. H. Schulzrunne, S. Casner, R. Frederick, and V. Jocobson, RTP: A Transport Protocol for Real-Time Applications (RFC 1889), IETF, 1996.
37. Chee-Wen Shiah, “The Application Sharing Technique and Its Utilization,” Ph.D. Dissertation, Department of Computer Science & Information Engineering, National Taiwan University, Taiwan, June 1997.
38. B. Su, “The Research and Implementation of the Shared Window System,” Master Thesis, Dept. of Computer Science & Information Engineering, National Taiwan University, Taiwan, June 1996.
39. Sun Microsystems Inc. Hot Java, <URL:http://www.sun.com/>.
40. SuperBrowser Project in Communications and Multimedia Lab., Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, <URL:http://cml11.csie.ntu.edu.tw/∼ssb>.
41. “The Java Language: An Overview,” <URL:http://www.javasoft.com/doc/Overviews/java/java-overview-l.html>
42. VDOnet Corporation, VDOLive Internet Video Servers and Players, <URL:http://www.vdolive.com/>, 1996.
43. Virtual Reality Modelling Language, <URL:http://weblynx.com.au/guide.htm>.
44. W.J. Welsh, “Model-based coding of videophone images,” Electronics and Communication Engineering Journal, pp. 29–36, 1991.
45. Workflow Management Coalition, From the Workflow Reference Model. Source: Speech by Nick Kingsbury at Groupware Europe 94, 1994.
46. World Wide Web Consortium, Hypertext Makeup Language, <URL:http://www.w3.org/hypertext/WWW/MakeUp>.
47. World Wide Web Consortium, Hypertext Transport Protocol, <URL:http://www.w3.org/hypertext/WWW/Protocols>.
48. World Wide Web Consortium (W3C) Activity: Audio and Video, <URL:http://www.w3.org/pub/WWW/Consortium/Prospectus/RealTime.html>, 1996.
49. World Wide Web Consortium (W3C) Activity: Collaboration, <URL:http://www.w3.org/pub/WWW/Collaboration/Overview.html>, 1996.
50. Xing Technology Corporation. StreamWorks, <URL:http://www.xingtech.com/>, 1996.
51. S.S. Yu, C.W. Shiah, L.H. Wang, and W.C. Chen, “Application sharing frameworks on java,” in Proc. 1997 International Conference on Consumer Electronics (ICCE ‘97), Chicago, USA, June 1997, pp. 14–15.



Ing-Chau Chang received his B.S. degree in Department of Computer and Information Science from National Chiao-Tung University, Hsin-Chu, Taiwan, R.O.C., in 1990. and the M.S. and Ph.D. degrees in Institute of Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in

1992 and 1997, respectively. He is currently an assistant professor in the Department of Information Management, ChaoYang University of Technology. His current research topics include computer-supported cooperative work, distance learning, multimedia network protocols, and multimedia systems.

Bo-Shen Liou was born in Taipei, Taiwan, R.O.C., on November 3, 1972. He received the B.S. degree in Computer Science & Information Engineering from the National Taiwan University, Taiwan, in 1995. He started his Ph.D. program since 1996, and he is a Ph.D. candidate in Computer Science & Information Engineering in the National Taiwan University, Taiwan. His current research interests are in the areas of high-speed computer networks and multimedia systems.

Jau-Hsiung Huang received the B.S. degree in Electrical Engineering from National Taiwan University, Taiwan, R.O.C. in 1981 and the M.S. and Ph.D. degrees in Computer Science from the University of California, Los Angeles, CA, U.S.A., in 1985 and 1988, respectively. Since 1988, he has been a member of the faculty in the Department of Computer Science and Information Engineering, National Taiwan University, where he is currently a professor. He co-founded the Communications and Multimedia Laboratory in the department in 1990 to launch series of research in the areas of distributed multimedia systems and virtual reality. He has published over 40 technical papers in the areas of multimedia networking, high-speed networking, parallel and distributed systems and performance evaluation of computing systems.

Shiuh-Sheng Yu received the B.S. degree in Computer Science and Information Engineering from National Taiwan University in 1989 and M.S. degree in Computer Science from SUNY at Stony Brook in 1992. He is

currently working towards the Ph.D. degree in Computer Science and Information Engineering at National Taiwan University. His research interests include multimedia information systems and distributed systems.

Chee-Wen Shia received his B.S. degree in Department of Computer and Information Science from National Chiao-Tung University, Hsin-Chu, Taiwan, R.O.C., in 1990 and the M.S. and Ph.D. degrees in Institute of Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1992 and 1997, respectively. His research interests include CSCW and multimedia systems.