Supporting Workspace-Mediated Interaction in Collaborative Presentations with CoPowerPoint

Supporting Workspace-Mediated Interaction in Collaborative Presentations with CoPowerPoint

Steven Xia
Griffith University
Brisbane, QLD 4111, Australia

David Sun
University of California at Berkeley
Berkeley, CA, USA

Chengzheng Sun
Nanyang Technological University
Singapore 639798

David Chen
Griffith University
Brisbane, QLD 4111, Australia

Abstract

Effective interaction among participants is crucial to the success of a presentation. The workspace of a presentation, which is the presentation slides, is a natural and effective medium of interactions among presentation participants. However, due to the insufficient support from existing collaborative presentation systems, workspace-mediated interaction is inconvenient in presentations, especially in collaborative presentations. In this paper, we explore typical workspace-mediated interaction forms and analyze their characteristics. Based on this analysis, we present a series of approaches to supporting workspace-mediated interaction in the context of the CoPowerPoint system.

1. Introduction

Presentations are widely employed in educational, academic, commercial, and technical settings for a variety of purposes, such as exchanging ideas, demonstrating products, and illustrating theories. Rapid advancements in software and hardware technologies have served to improve and enhance the quality and convenience of presentations. An important type of presentation software systems is the collaborative presentation (also known telepresentation [1]) systems, which support presenting to remote audiences via computer networks. With the support of these systems, participants are free to attend a presentation locally or remotely. For remote audience, without the need for physical presence at where the presentation is delivered, significant amount of time and cost involved in traveling are saved. Hence, collaborative presentations have become not only an attractive alternative to physically attending face-to-face presentations, but also a more flexible and convenient means of presentation delivery and attendance [3].

Effective communication and interaction among the participants is particularly important to achieving high degree of success in the delivery of presentation of any form. By interacting with audiences, a speaker can obtain valuable feedback that allows him/her to make suitable adjustments to the content and the rhythm of delivery. From interacting with the speaker, audiences can attain a better understanding of the ideas and issues the speaker tries to convey in the presentation. Moreover, from interacting with each other, audiences can create an atmosphere of active participation, which helps to focus their attention on the presentation [2]. Effective interaction among participants plays an even more important role in computer-supported collaborative presentations in the sense that effective interactions are able to reduce the distance among distributed participants.

Slides are the central medium to convey the presentation content and thus are also the central medium for some interaction forms, such as questioning and discussion. For example, in a presentation about a blueprint of a computer network, an audience may ask questions about the network architecture demonstrated in a slide, e.g. “what is the purpose of that switch”. In discussion, the audience may propose alternative solutions to improve the design presented in a slide, such as “what if we add a bridge to link these two networks”. Compared with implicit interactions such as body gestures or eye contacts between the speaker and audience, the most significant characteristic of these interaction forms is that they are mediated by the presentation slides. Since the presentation slides are used as the main workspace of a presentation, we use the term *workspace-mediated interaction* to denote this kind of interactions.

Although workspace-mediated interaction is playing a particularly important role in presentations, existing presentation software systems have hardly provided any supports to them. With this functional insufficiency, workspace-mediated interaction is often inconvenient. As a result, the following scenarios are frequently observed in presentations. In the questioning stage of our example presentation mentioned above, an audience has a question about the function of a device displayed in a slide. To raise the question, the audience has to ask the speaker to turn the slides back and forth to find the slide containing

1-4244-0030-9/05/$20.00 ©2005 IEEE

the materials related to the question. Being unable to indicate the exact slide number, it takes time to finish this step. Then the audience needs to further locate objects related to the question in that slide. If that slide contains quite a few object of the same type, locating the correct one is not easy either. Usually, the audience has to fall back on ambiguous descriptions, such as "the red rectangle a litter right to the center". These troubles can be even worse in discussions. In the same example, when an audience proposes an improvement to the design by adding a bridge at a specific location, it may be difficult for the speaker and other audience to find the location based on such ambiguous descriptions. For the remote audience attending the presentation with a collaborative presentation system, the physical distance would further aggravate this situation.

The root of this problem is that the audience is excluded from the control of slides. Consequentially, when they need to refer to slide contents, they have to adopt some indirect means (e.g. oral descriptions).

To address this problem, we present novel approaches in this paper to supporting workspace-mediated interaction in collaborative presentations. The basic idea is to enable both local and remote audiences with the same slides manipulate capabilities as the speaker dose, such as changing the current presented slide, making annotations and editing the slide content, so that they are able to directly manipulate the slides in workspace-mediated interaction. To verify the correctness and effectiveness, we have implemented these approaches in the CoPowerPoint[1] system.

The rest of this paper is organized as follows. First, interaction supporting features of existing collaborative presentation systems are summarized in Section 2. Afterwards, typical workspace-mediated interaction forms and requirements for supporting them are discussed in Section 3. Next, the CoPowerPoint system is introduced in Section 4. In Section 5, approaches for supporting workspace-mediated interaction are presented in the context of the CoPowerPoint system. Finally, this paper concludes with the contribution and future work in Section 6.

2. Related Work

As the result of active research on presentation supporting techniques, there exist numerous of collaborative presentation software systems. Apart from providing basic supports to presentations (e.g. video, audio and slides broadcasting), these systems also strive to support and enhance interactions from different angles.

TELEP [4] system is designed to deliver presentations to both local and remote audience at the same time. It enhances interactions among speakers, local and remote audiences by displaying audience images on the user interface. It also supports the communication between the speaker and remote audience with a back voice channel and a text chatting tool.

Similar to TELEP, Forum [2] also supports the back voice channel and text chatting. Moreover, Forum provides more interaction features, including a poll meter, with which the speaker can initiate a vote, and an annotation tool, with which the speaker can broadcast annotations on slides to all audience. Meanwhile, the audience is allowed to perform some basic slide manipulating operations, including browsing or annotating their local slide copies.

[5] represents remote audience in a virtual space and integrates the virtual space with the physical space of the local audience so that the two spaces seem to be adjacent but distinct components in a single space. This integration increases the awareness among the speaker, local and remote audiences, and helps to enhance interactions among participants.

These systems focus on supporting non-workspace-mediated interaction. To the best of our knowledge, no existing collaborative presentation systems have provided any supports to workspace-mediated interaction. Therefore, when workspace-mediated interaction occurs in a presentation, the inconvenience problems cannot be solved with existing collaborative presentation systems.

3. Requirements for Supporting Workspace-Mediated Interaction Forms

Before designing approaches to supporting workspace-mediated interaction, we shall discuss basic requirements for a collaborative presentation system to support workspace-mediated interaction.

3.1. Functional Requirements

In this subsection, we shall enumerate workspace-mediated interaction forms frequently used in presentations and discuss functional requirements for supporting these interaction forms respectively.

3.1.1 Questioning. Questioning is a common interaction form in presentations. Most presentations end up with a questioning process. In a questioning process, the audience raises doubts or unclear issues and the speaker gives answers or clarifications in response. Questioning not only solves the audience's problems in understanding the presentation, but also helps the speaker learn the audience's interests and hence improve the presentation.

[1] CoPowerPoint Demo. http://reduce.qpsf.edu.au/copowerpoint.

Questioning is a workspace-mediated interaction. When an audience raises a question, he/she may refer to the content in a specific slide. So does the speaker while answering a question. To facilitate questioning, a collaborative presentation system needs to provide the following functionalities.

1. *Free Slide Browsing*. An audience should be allowed to view any slides other than the current presented one. With this functionality, the audience is able to browse presentation slides when preparing the question.
2. *Current Presented Slide Selecting*. An audience should also be allowed to select the current slide presented to all participants. With this functionality, the audience can directly bring all participants to the slide containing materials related to the question.
3. *Slide Annotating*. An audience should be allowed to make annotations on the current presented slide. Furthermore, the annotations should be displayed on all participants' screens. With this functionality, the audience can precisely highlight objects in a slide without resorting on ambiguous descriptions.
4. *Collaborative Slide Editing*. An audience should be allowed to edit the content of the current presented slide. This functionality help the audience in raising questions related to alternative ideas to the presented ones, such as "why not add this object in the system". Moreover, while editing the current presented slide, the questioning audience may also invite other audience with expertise in the presentation topic to collaboratively edit the slide. This requires the slide editing functionality to be collaborative.
5. *Speaker-Coordinated Floor Control.* The questioning process is coordinated by the speaker. To enforce the interaction order, the collaborative presentation system should provide a speaker-coordinated floor control mechanism to circulate the floor between the speaker and audience.

3.1.2. Discussion. Discussion has a higher interactivity than questioning. In discussion, participants propose or defend their opinions on specific topics. As a multi-round interaction form, discussion is used to clarify complex issues and find solutions to complex problems. It also gives the audience an opportunity to contribute their knowledge to the presentation topics. Furthermore, the active involvement in discussion enhances the audience's understanding and memorizing. Therefore, discussion is particularly important in educational presentations. In symposia and panel discussions, discussion is the major interaction form.

Discussion is also a workspace-mediated interaction, because discussed topics in a presentation are often related to the content presented in slides. The functionalities needed for supporting questioning are also needed for supporting discussion.

In discussion, all participants are free to take turns to speak. Without the coordination of the speaker, discussion is a more free interaction form than questioning. To support the flexibility of discussion, the following functionalities are needed.

1. *Free Floor Control.* Unlike questioning, all participants in a discussion process are equal. There is no dominator. So, the collaborative presentation system should provide a free floor control mechanism to circulate the floor among all participants.
2. *Slide Locking.* When a participant is speaking, other participants may be editing some slides to prepare their proposals. To avoid interfering the speaking participant, other participants that are invited to collaborate with the speaking participant should be prohibited from editing the current presented slide. To achieve this, a locking mechanism is needed to lock the current presented slide.

3.1.3. Group Discussion. Group discussion is a special form of discussion. In this interaction form, participants are divided into groups. Members of the same group view the same presented slide. Annotations are only visible to group members. Only one participant in a group is allowed to speak at a time. Given these characteristics, the functionalities needed for supporting discussion are also needed for supporting group discussion, but the scope is limited within a group.

The speaking participants from multiple groups may attempt to lock slides simultaneously. Locking attempts may conflict with each other if they target on the same slide. The collaborative presentation system needs to extend the slide locking mechanism for supporting discussion to resolve locking conflicts.

3.2. The Architectural Requirement

To support workspace-mediated interaction, the collaborative presentation system needs to adopt a symmetric distributed architecture, which is shown in Figure 1.

In this symmetric distributed architecture, all participants in the presentation are supported by the same presentation software system, with the same slide manipulation functionalities that meet the functional requirements discussed in Section 3.1.

This architecture effectively facilitates supporting workspace-mediated interaction. When a participant manipulates the presentation state in the local system, corresponding operations will be generated. These operations are propagated to and executed at remote systems. In this way, every participant is able to change

the presentation state of other participants, like the speaker does.

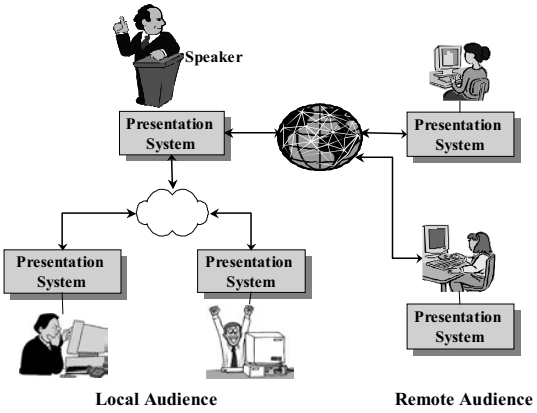

Figure 1. The symmetric distributed architecture.

Apart from the ability to support workspace-mediated interaction, another advantage of this architecture is that we do not need to develop different systems for the speaker and audience respectively. Instead, a single system can meets the functional requirements of all participants. This significantly reduces the designing and implementation efforts.

In contrast, existing collaborative presentation systems are designed in an asymmetric distributed architecture, as shown in Figure 2.

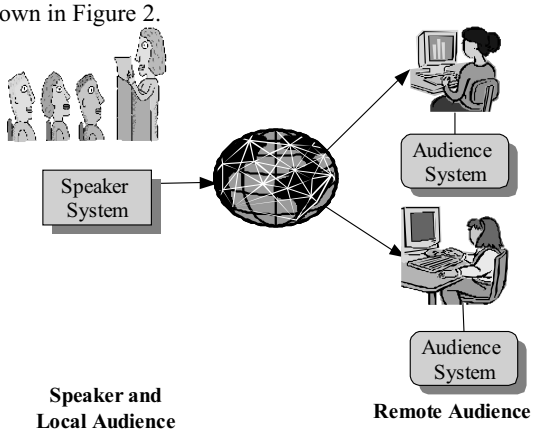

Figure 2. The asymmetric distributed architecture.

In this asymmetric distributed architecture, the speaker and audience are supported by different software systems. Only the speaker has the capability of manipulating the presentation state. Presentation operations unilaterally flow from the speaker system to the audience systems, and the audience systems passively receive and execute these operations. Furthermore, in some systems, the slide copies in the speaker system and the audience system are in different formats in the sense that the speaker's copy is editable but the audience's copies are not. Finally, in most existing collaborative presentation systems, the local audience is not supported by software systems. They simply view the slides projected on a big screen.

In this symmetric distributed architecture, only the speaker system has the functionalities that meet the functional requirements discussed in Section 3.1, so it is clear that this architecture cannot support workspace-mediated interaction.

Based on a collaborative presentation system that meets the requirements discussed in this section, approaches can be designed to support workspace-mediated interaction. In the remaining part of this paper, we shall discuss these approaches in the context of a collaborative slide authoring and presentation system, CoPowerPoint.

4. The CoPowerPoint System

In this section, we shall take the CoPowerPoint system as an example to present approaches to supporting workspace-mediated interaction in collaborative presentation systems.

4.1. Introduction to CoPowerPoint

CoPowerPoint is a collaborative slide authoring and presentation system, which is converted from MS PowerPoint[2] with the *Transparent Adaptation* (TA) [9] approach. With the TA approach, the original PowerPoint is used as the user interface. The user's local input events are intercepted before they reach PowerPoint; executed at the local site immediately to achieve high responsiveness; and then propagated to remote sites and replayed there under the control of underlying collaboration supporting techniques. The input events interception, propagation, and replay are all achieved by means of the Application Programming Interface (API) of the single-user application (PowerPoint) and its execution environment (Windows), without access to the application's source code.

The key to achieving unconstrained collaboration is the adaptation of the single-user PowerPoint API to meet the data and operational modeling requirements of the underlying foundation collaboration technique *Operational Transformation* (OT) [7][8]. The reason for choosing OT as the foundation collaboration technique is that OT is the state-of-the-art unconstrained collaboration technique for supporting consistency maintenance (for

[2] Microsoft PowerPoint. http://www.microsoft.com/powerpoint.

group do) and group undo in a wide range of editor-like applications.

CoPowerPoint adopts a replicated architecture, in which each user runs an independent CoPowerPoint instance in the local system. The shared document (i.e. the presentation slides) is also replicated in each CoPowerPoint instance. All CoPowerPoint replicas are connected to a central Session Manager, which provides CoPowerPoint instances with a series of services including document access, session management, etc..

4.2. Multi-User Interfaces and Functionalities of CoPowerPoint

CoPowerPoint converts the single-user interfaces and functionalities of PowerPoint into multi-user counterparts. To understand the multi-user interfaces of CoPowerPoint, it is necessary to have a clear understanding of single-user interface and functionalities of PowerPoint first.

PowerPoint provides the user with multiple levels of interface, called views, to edit data objects in a document. A top level editing interface is *slide sorter view*. In this view, a PowerPoint document is presented as a sequence of slides, and the granularity of the user's actions is at the slide level. For example, the user can insert or delete slides, re-arrange the order of slides, or customize the theme or background of slides. The contents of individual slides cannot not be directly manipulated by the user at this level.

From the *slide sorter view* interface, the user can "zoom" into any individual slide to edit the contents of that particular slide. The editing interface at this level is called *normal view*. The *normal view* interface includes three editing panes. The first pane is the *slide pane*, which is the same as any graphic editing application, where graphic objects can be created at arbitrary positions on a drawing canvas and moved freely. From this pane, the user can create, remove, or change any data objects, including text-boxes, images, clip-arts, drawings, etc.. The second pane is the *slide notes pane*, which is a text editing area for the user to write explanatory notes for the corresponding slide. The last pane is the *outline pane*, which provides an outline of all slides in the document. In this pane, the user can perform some restricted functions available in the *slide sorter view* (e.g., creating or deleting slides).

Apart from these editing views, there is another presentation interface, called *slide show view*. From this interface, the user can control the presentation (e.g., go to the next, previous, or a specific slide, animation, or annotate the presentation screen with a virtual pen, etc.), but cannot change the contents of slides.

Powered by the TA approach, the above single-user interfaces are preserved and single-user functionalities are converted into multi-user ones. CoPowerPoint is able to support unconstrained collaborative slide editing in the above views. Particularly, the user can collaboratively edit the slide sequence from the *slide sorter view*, edit objects in a slide and slide notes from the *normal view* and view the presentation in the *slide show view*.

These comprehensive multi-user interfaces and functionalities are the important foundations for the CoPowerPoint system to support collaborative presentation and workspace-mediated interaction, as will be illustrated in the following subsections.

4.3. Interfaces and Functionalities of CoPowerPoint for Supporting Workspace-Mediated Interaction

To run a presentation with CoPowerPoint, all users should be in the slide-show view. In this view, the speaker uses the presentation functionalities of PowerPoint to control the presentation. When the speaker selects a slide as the current presented one, the speaker's CoPowerPoint system generates an operation carrying the identifier of the current presented slide is generated and propagates the operation to all audience systems. After receiving this operation, an audience's CoPowerPoint system executes it and displays the specified slide in the *slide show view*. When the speaker draws a line on the current presented slide as an annotation, the speaker's CoPowerPoint system generates an operation carrying the color and position information of the line and propagates the operation to all audience systems. After receiving this operation, an audience's CoPowerPoint system draws the same line on the *slide show view*.

To support the lecturing mode in which only the speaker has the exclusive control of the presentation and the audience is prohibited from generating presentation operations, all audience inputs to the *slide-show view* of PowerPoint that could change the presentation states are intercepted and discarded.

To allow a user to manipulate the presentation slides (e.g. when an audience is holding the control while raising a question), his/her inputs to the *slide show view* are passed to PowerPoint and executed locally. Corresponding presentation operations are generated and propagated to and executed at remote CoPowerPoint systems.

To support workspace-mediated interaction, the collaborative presentation system needs to allow users to view and edit the presentation slides from different interfaces. This can be easily supported in CoPowerPoint thanks to the exiting single-user interfaces of PowerPoint and the *data-centric* property of the TA approach.

To support free slide browsing, a user can switch to the *slide sorter view*, in which all slides are listed in miniatures. In this view, the user can browse all

presentation slides. Moreover, the user can also select a slide and then switch to the *normal view*, in which details of this slide are displayed. To prohibit the user from editing the slide contents, all inputs to PowerPoint interface that could modify the slides are intercepted and discarded.

If slide editing is allowed (e.g. for the speaking participant in a group discussion), users can also collaboratively edit the slide content in the *slide sorter* or *normal views*. The consistency of slide content in the face of collaborative editing is guaranteed by the TA approach and underlying OT mechanism.

When a user is not in the *slide-show view*, presentation operations received from remote sites should not be executed. This is because when the user is not viewing the current presented slide, the current slide should not be changed by the speaker. Annotations should not be drawn either since the user is viewing a different slide. Moreover, when the speaker is not in the *slide-show view*, presentation operations should be generated either.

5. Supporting Workspace-Mediated Interaction in CoPowerPoint

Based on the multi-user interface features and functionalities of CoPowerPoint, we shall discuss approaches for supporting the workspace-mediated interaction forms in this section.

5.1. Supporting Questioning

The nature of questioning is that the speaker and audience take turns to manipulate the presentation slides. So, a floor control based mechanism is suitable for this interaction form. Moreover, this floor control is coordinated by the speaker, because (1) before any audience asks for the control, the speaker holds it; and (2) after an audience finishes asking a question, the control should go back to the speaker, rather than going to another audience.

The questioning process is shown in Figure 3. First, the audience may browse presentation slides while preparing the question. Then, the audience expresses the intension to raise a question by sending a question control request to the speaker. After receiving the request, the speaker grants the control to the audience and sends an approval to the requesting audience. After obtaining the control, the audience is able to manipulate the presentation slides, including changing the current presented slide, annotating or editing the current presented slide. After finishing raising the question, the audience returns the control to the speaker, and the speaker can manipulate the presentation slides while answering the question.

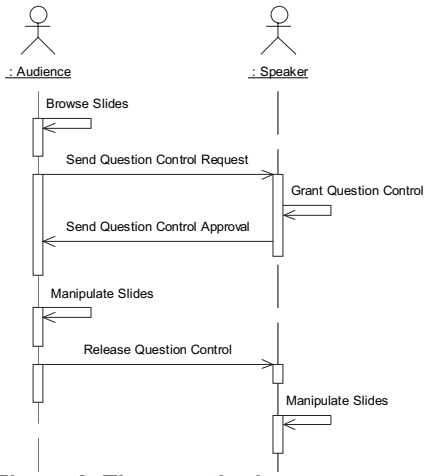

Figure 3. The questioning process.

Multiple audience may send control requests to the speaker simultaneously. To handle the simultaneous requests, the speaker system needs to maintain a request queue. When a request is received at the speaker site, if the control is available, it is granted to the requesting audience immediately. Otherwise, the request is buffered in the request queue, as shown in the routine in Figure 4.

```
HandleQuestionControlRequest(req)
{
   if(control_aid = = null)
      GrantQuestionControl(req);
   else
      Append(request_queue, req);
}
```

Figure 4. The routine for the speaker to handle a control request.

After finishing answering a question, the speaker may process the requests buffered in the request queue and grant the control to the next audience, as shown in Figure 5.

```
HandleQueuedQuestionRequest(req)
{
   if(!request_queue.empty)
      GrantQuestionControl(GetNextReq(request_queue));
   else
      control_aid = null;
}
```

Figure 5. The routine for the speaker to handle a control release message.

To grant the question control to an audience, the speaker system sets the controlling audience identifier and sends a response message to the requesting audience. The *GrantQuestionControl* routine used in the above routines is sketched in Figure 6.

```
GrantQuestionControl(req)
{
    control_aid = req.aid;
    SendMsg(req.aid, ControlApproval);
}
```

Figure 6. The routine for the speaker to grant the control to an audience.

5.2. Supporting Discussion

In a discussion process, all participants are equal and there is no coordinator like the speaker in the questioning process. In this case, the floor management can be performed by another dedicated coordinator or in a distributed way [6]. The coordinator-based mechanism is simpler and involves fewer messages, because it requires only a request and a response between the requester and the coordinator. On the contrary, distributed algorithm like [6] requires messages to be broadcast to all participants. Furthermore, in the CoPowerPoint system, there exists a central Session Manager, which is ideal to play the role of the coordinator. Therefore, we have taken the coordinator-based approach. With this approach, before a participant speaks, he/she must apply for the control from the Session Manager. The process of discussion is shown in Figure 7. Before speaking, a participant may browse or edit existing slides to prepare the proposal. Then the participant expresses the intension to speak by sending a discussion control request to the Session Manager. After receiving this request, the Session Manager grants the participant with the control by sending back an approval of the request. While speaking, the participant is able to generate presentation or editing operations. After finishing speaking, the participant releases the control.

To handle simultaneous control requests from multiple participants, the Session Manager also maintains a request queue. Requests in the queue are processed according to their arriving order, as sketched in the routine for supporting questioning in Section 5.1. The only difference is that the Session Manager processes queued requests every time when a participant releases the control.

After obtaining the control, a participant may invite other participants to collaboratively edit the currently presented slide to help proposing ideas. At the same time, other participants may concurrently edit other slides for preparing their proposals, but should be prohibited from editing the current presented slide to avoid interference. Therefore, the presentation of every participant should always record the identifier of the current presented slide and prevent the local participant from editing this slide without the control.

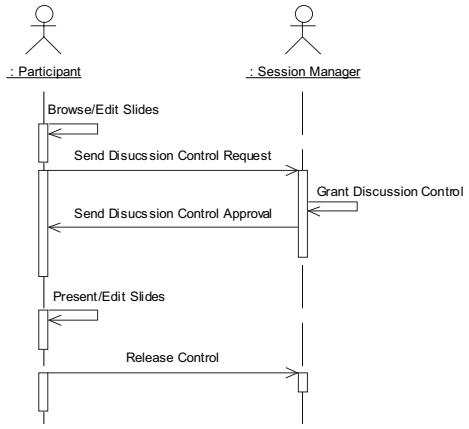

Figure 7. The discussion process.

5.3. Supporting Group Discussion

Like in a discussion process, a participant in a group discussion needs to lock the current presented slide for exclusive editing. This situation introduces a two-level competition among participants. Group members may compete for the control within a group; and groups may compete for the lock of a common slide. The former competition can be solved with the same mechanism for supporting discussion, but the scope is limited within individual groups. The latter requires a conflict resolution mechanism.

The exclusive nature of locking allows only one participant to lock a slide. For participants that have failed in conflicts, one possible strategy is to deny their access to the target slide. An alternative is to allow them to access the target slide in a read-only mode (i.e. they are allowed to choose the target slide as the current presented slide and annotate it, but are not allowed to edit it). Furthermore, if the target slide is unlocked, one of the participants failing to lock it is granted with the lock. In our system, we have adopted the latter approach because it minimizes the impact of the locking conflicts.

To achieve this effect, the Session Manager maintains the following two tables.

1. *Group Control Table* (GCT). Every item in this table corresponds to a discussion group. An item in this table contains the identifier of the participant that is holding the control and a queue of pending requests for the control of this group.
2. *Slide Lock Table* (SLT). Every item in this table corresponds to a slide. An item in this table contains the identifier of the participant that has successfully locked the corresponding slide and a queue of requests that have failed to lock this slide.

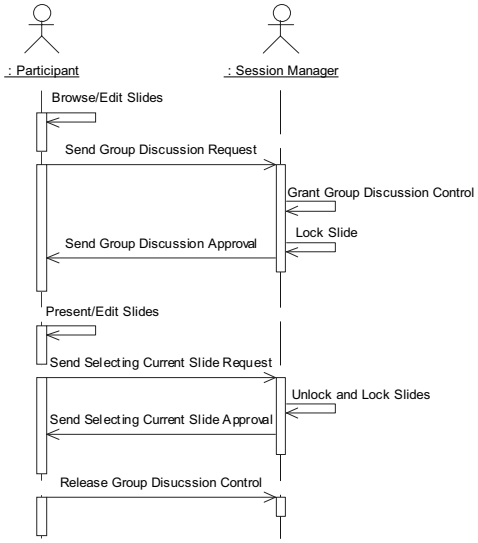

Figure 8. The group discussion process.

The process of group discussion is shown in Figure 8. Similar to the discussion process, before a participant speaks, he/she may browse or edit existing slides. Then a *group discussion control request* is sent to the Session Manager to indicate the intension of speaking. After receiving this request, the Session Manager checks the GCT to see whether the control of the corresponding is available. If so, the control is granted to the requesting participant immediately. Otherwise, the request is buffered in the request queue for this group, as s shown in the routine in Figure 9.

```
HandleGroupDiscussionControlRequest(req)
{
   if(GCT[req.gid].control_pid = = null)
      GrantGroupDisucssionControl(req);
   else
      Append(GCT[req.pid].request_queue, req);
}
```

Figure 9. The routine for the Session Manager to handle a group discussion control request.

After receiving the approval, the participant is able to manipulate the presentation state or editing the slide content. Moreover, when the participant is speaking, multiple slides may be referred. When the participant attempts to select another slide as the current presented one, a *selecting current slide request* is sent to the Session Manager.

The routine for handling this request is shown in Figure 10. After receiving this request, the Session Manager unlocks the current presented slide of this group by calling the *UnlockSlide* routine, and tries to lock the specified new slide by calling the *LockSlide* routine. The *LockSlide* routine returns the access mode (e.g. read-only or read-write) to the target slide allowed for the requesting participant. Finally, a response is sent back to the requesting participant together with the access mode.

```
HandleSelectingCurrentSlide(req)
{
   UnlockSlide(req);
   access_mode = LockSlide(req);
   SendMsg(req.pid, ChangeSlideApproval, access_mode );
}
```

Figure 10. The routine for the Session Manager to handle the *selecting current slide request* in group discussions.

When a participant has finished speaking, a *group discussion control release* message is sent to the Session Manager. After receiving this message, the Session Manager unlocks the current presented slide used by this participant. If the request queue of this group is not empty, the control is granted to the first requesting participant in the queue. Otherwise, the control is marked available for any member of this group to apply for, as shown in Figure 11.

```
HandleGroupDiscussionControlRelease (req)
{
   UnlockSlide(req);
   if(!GCT[req.gid].request_queue.empty)
   {
      queued_req = GetNextReq(GCT[req.gid]);
      GrantGroupDisucssionControl(queued_req );
   }
   else
      GCT[req.gid].control_pid = null;
}
```

Figure 11. The routine for the Session Manager to handle group discussion control release message.

The *GrantGroupDiscussionControl* routine used in the above routines is shown in Figure 12. To grant the group discussion control to a participant, the Session Manager first marks the corresponding item in the GCT as being controlled by the requesting participant. Then it tries to lock the specified slide and gets the allowed access mode. Finally, it sends a response message back to the requesting participant.

```
GrantGroupDisucssionControl(req)
{
   GCT[req.gid].control_pid = req.pid;
   access_mode = LockSlide(req);
   SendMsg(req.pid, ControlApproval, access_mode);
}
```

Figure 12. The routine for the Session Manager to grant the group discussion control to a participant.

When a slide is unlocked, one of the participants that have failed to lock this slide should be granted with the lock. To achieve this effect, the *LockSlide* and *UnlockSlide* in the above routines are sketched in the following figures.

```
LockSlide(req)
{
   if(SLT[req.slide_to_lock] = = null)
   {
      SLT[req. slide_to_lock].lock_pid = req. pid;
      return ReadWrite;
   }
   else
   {
      Append(SLT[req. slide_to_lock].fail_queue, req);
      return ReadOnly;
   }
}
```

Figure 13. The routine for the Session Manager to lock a slide.

The *LockSlide* routine is shown in Figure 13. If the requested slide is not locked, the corresponding item in SLT is marked as being locked by the requesting participant. In this case, the requesting participant can access the requested slide in a read-write mode. If the requested slide has already been locked previously, the identifier of the requesting participant is buffered in a failing queue of this slide, and the requesting participant is only allowed to access the requested slide in a read-only mode.

```
UnlockSlide(req)
{
   if(SLT[req.slide_to_unlock].locking_site = = req.pid)
   {
      if(! SLT[req. slide_to_unlock].fail_queue.empty)
      {
         fail_req =
            GetNextReq(SLT[req. slide_to_unlock].fail_queue);
         SLT[req. slide_to_unlock].lock_pid=fail_req.pid;
         SendMsg(fail_req.pid, PermissionUpdate, ReadWrite);
      }
      else
         SLT[req.slide_id].lock_pid = null;
   }
   else
      Remove(SLT[req. slide_to_unlock].fail_queue, req.pid);
}
```

Figure 14. The routine for the Session Manager to unlock a slide.

The *UnlockSlide* routine is shown in Figure 14. If the requesting participant is the one that is holding the lock of the target slide and the failing queue of the target slide is not empty, the first participant in the failing queue is granted with the lock and read-write access permission to this slide. Then, this participant is notified about the permission update. If the failing queue is empty, this slide is marked as unlocked. If the requesting participant is not the lock holder of the target slide, it must be one of the failing participants. In this case, its locking request is removed from the failing queue.

6. Conclusion and Future Work

In this paper, we have contributed novel approaches to supporting workspace-mediated interaction in collaborative presentation software systems. We have discussed the importance of supporting workspace-mediated interaction, analyzed the characteristics of each workspace-mediated interaction form and requirements for supporting workspace-mediated interaction. We have also presented approaches to supporting these interaction forms in the context of the CoPowerPoint system.

It is important for collaborative presentation systems to provide sufficient support to interactions. In the future, we plan to investigate other interaction forms involved in presentations and design corresponding supporting techniques. Moreover, we also plan to extend the CoPowerPoint system with other frequently used functionalities such as audio and video supports.

7. Reference

[1] Gemmel, D.J., Bell, C.G., "Noncollaborative telepresentation comes to the age", *Communication of the ACM*, 40(4), April 1997, pp. 79-89.

[2] Isaacs, E.A., Morris, T., and Rodriguez, T.K., "A forum for supporting interactive presentations to distributed audiences", *In Proceedings of ACM conference on Computer-Supported Cooperative Work*, 1994, pp. 405-416.

[3] Isaacs, E.A., Morris, T., Rodriguez, T.K., & Tang. J.C.: "A Comparison of Face-to-Face and Distributed Presentations", *In Proceedings of the Conference on Computer-Human Interaction,* 1994, pp. 354-361.

[4] Jancke, G., Grudin, J., and Gupta. A., "Presenting to local and remote audiences: design and use of the TELEP system", *In Proceedings of the SIGCHI conference on Human factors in computing systems,* 2000, pp. 384-391.

[5] Koleva, B., Schnadelbach, H., Benford, S., and Greenhalgh, C.: "Experiencing a presentation through a mixed reality boundary", *In Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, 2001, pp. 71-80.

[6] Ricart, G., Agrawala, A.K., "An optimal algorithm for mutual exclusion in computer networks",

Communications of the ACM, 24(1), January 1981, pp. 9-17.

[7] Sun, C. and Ellis, C.A.: "Operational transformation in real-time group editors: issues, algorithms, and achievements", *In Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 1998, pp. 59-68.

[8] Sun, C., Jia, X., Zhang, Y., Yang, Y., and Chen, D.: "Achieving convergence, causality-preservation, and intention-preservation in real-time cooperative editing systems", *ACM Transactions on Computer-Human Interaction,* 5(1), March 1998, pp. 63-108.

[9] Xia, S., Sun, D., Sun, C., and Chen, D.: "Leveraging single-user applications for multi-user collaboration: the CoWord approach", *In Proceedings of ACM Conference on Computer-Supported Cooperative Work*, November 2004. pp. 162-171.