Supporting Worker Independence in Collaboration Transparency

James "Bo" Begole *Mary Beth Rosson* *Clifford A. Shaffer*
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061
{begolej, rosson, shaffer}@cs.vt.edu

ABSTRACT

Conventional collaboration-transparency systems are inefficient in their use of network resources and lack support for key groupware principles: concurrent work, relaxed WYSIWIS, and group awareness. We present an alternative implementation approach to collaboration transparency that provides many features previously seen only in collaboration-aware applications. Our approach is based on an object-oriented replicated architecture where selected single-user interface objects are dynamically replaced by multi-user extensions. The replacement is transparent to the single-user application and its developer. As an instance of this approach, we describe its incorporation into a new Java-based collaboration-transparency system, called Flexible JAMM.

We conducted an emperical study to evaluate the effectiveness of Flexible JAMM versus a representative conventional collaboration-transparency system, NetMeeting. Completion times were significantly faster in a loosely-coupled task using Flexible JAMM, and were not adversely affected in a tightly-coupled task. Accuracy was unaffected by the system used. Participants greatly preferred Flexible JAMM. The evaluation validates our aim of supporting multiple styles of collaboration.

KEYWORDS: Computer-supported cooperative work, groupware, collaboration transparency, usability, Java.

INTRODUCTION

There are two general approaches to providing computer support for synchronous collaboration. One is the sharing of legacy single-user applications. This is called **collaboration transparency,** because the sharing is provided by a mechanism that is unknown, or transparent, to the application and its developers. The second approach, **collaboration awareness,** is to design an application specifically to support cooperative work.

Several collaboration-transparency systems now exist, particularly for the X Window System. Examples include Old Dominion University's XTV [1], Hewlett Packard's SharedX [8] and Sun Microsystem's ShowMe SharedApp [16]. Recently, Microsoft has released NetMeeting, a freely-available collaboration-transparency system for the Windows platform [13]. We refer to these and other currently available collaboration-transparency systems as **conventional,** because they share the same basic implementation and resulting limitations, as described later in this section. In contrast, we discuss a **flexible** collaboration-transparency system in the next section.

In terms of development cost, conventional collaboration transparency has the advantage over collaboration awareness (at the application level, conventional collaboration transparency has no development cost). However, there are differences in the quality of support for collaboration that these types of systems provide. The degree to which collaborators work closely together versus independently is referred to as **tight** versus **loose coupling** [6]. In contrast to some collaboration-aware applications, conventional collaboration-transparency implementations impose an inflexible, tightly-coupled style of collaboration and do not adequately support key groupware principles [2]:

Concurrent work Conventional collaboration-transparency systems do not allow simultaneous input from more than one person.

WYSIWIS In conventional collaboration-transparency systems, all participants see exactly the same view at the same time in a manner referred to as strict **What You See Is What I See (WYSIWIS)** [15].

Group awareness Because they use strict WYSIWIS, conventional collaboration-transparency systems indicate remote users' cursor locations by means of **telepointers.** However, telepointers are inadequate to convey users' locations when WYSIWIS is relaxed [9, 14].

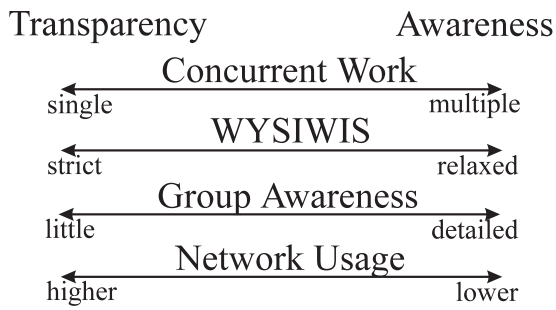

Figure 1: Four dimensions along which convential collaboration-transparency systems and collaboration-aware applications differ.

Network usage Primarily resulting from the use of a centralized display broadcasting architecture, conventional collaboration-transparency systems generally require higher network bandwidth than collaboration-aware applications.

Figure 1 illustrates where the two types of systems lie along these dimensions. It is important to note that while conventional collaboration-transparency systems are clustered at the left end of each dimension, collaboration-aware applications range up and down the scales; a particular collaboration-aware application may be no better than conventional collaboration transparency in one or all dimensions.

The remainder of this paper consists of two parts. First, we describe an approach to collaboration transparency that provides many features previously seen only in collaboration-aware applications. Our approach is based on a replicated architecture under which selected single-user interface objects are dynamically replaced with multi-user extensions. Transparency is maintained, because the original application source code is unmodified. We illustrate the approach with examples of its incorporation in a replicated collaboration-transparency system, called Flexible JAMM (Java Applets Made Multiuser), which is a new system extending the architecture of our previous work by now providing flexible multi-user support [3, 4]. Second, we describe an emperical study that evaluated the effectiveness of our flexible collaboration-transparency system versus a representative conventional collaboration-transparency system, NetMeeting, for performing tightly- and loosely-coupled tasks.

FLEXIBLE COLLABORATION TRANSPARENCY

Despite the problems previously enumerated, collaboration transparency remains attractive because of the low development cost of sharing legacy applications. We propose a way to add support of groupware principles to collaboration transparency. Our approach is to combine a replicated architecture and dynamic replacement of particular single-user interface objects with multi-user extensions. This section describes the replicated architecture and object replacement technique and the advantages they afford. We illustrate the approach by describing its incorporation into an instance of a replicated-architecture collaboration-transparency system called JAMM (Java Applets Made Multiuser, see also [4]) resulting in a new system called Flexible JAMM.

Replicated Architecture

Our approach is based on a replicated architecture, where each collaborator maintains a copy of the shared application. In prior work, we found that current technology can support a replicated architecture for transparent sharing of many, though not all, single-user applications [4, 12]. In addition to allowing lower network bandwidth usage, replication facilitates support of concurrent work and relaxed WYSIWIS, as described later in this section.

Dynamic Object Replacement

Today, the user interface of an object-oriented application is typically constructed by combining graphical user interface (GUI) objects. As useful interface objects become popular, application platforms provide them as standard components. Examples include the classic Windows, Icons, Menus, Pointers, and Scrollbars (WIMPS). Many compound interface components are also commonly available, such as scrollable lists, editable text areas, and dialog windows.

Polymorphism In our approach to flexible collaboration transparency, particular standard single-user interface components are replaced by multi-user equivalents at the time the application is initially shared. This scheme depends on the polymorphism of an object-oriented platform, such as Smalltalk and Java. Polymorphism allows an instance of one class to function in place of another class, provided the replacement has an object interface that is signature-equivalent to the original. Thus, an instance of a single-user interface class may be replaced by an instance of a polymorphically-equivalent multi-user class. Since the application source code is not modified, we consider such substitution to be *transparent* to the shared application.

Semantic Knowledge For those interface objects that it replaces, the flexible collaboration-transparency system has knowledge of the semantics of input events that occur within that object. With this information, the system can optimize its use of network resources by minimizing the size and number of update messages distributed to replicas. This semantic knowledge also allows a flexible collaboration-transparency system to provide relaxed WYSIWIS, implicit floor control, concurrent work, and fast local application response. The remainder of this section describes our implementation

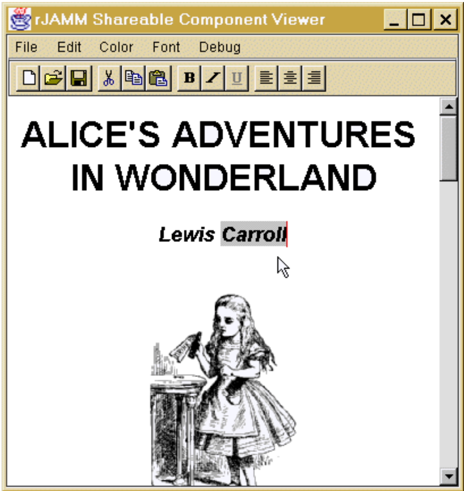

Figure 2: A single-user text editing application, Stylepad, that contains a scrollable panel as a main interface component.

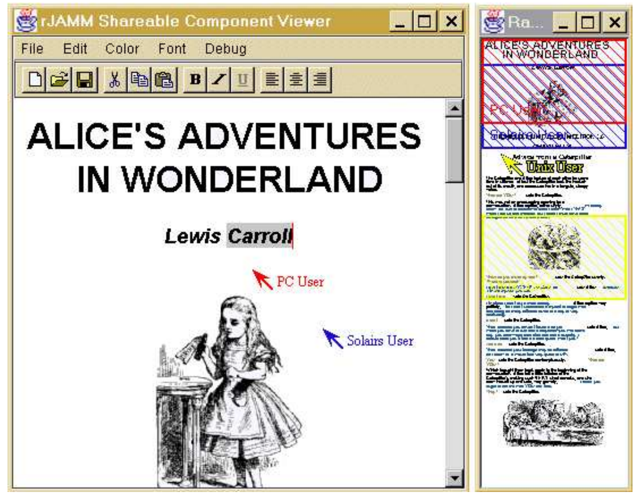

Figure 3: Stylepad shared via Flexible JAMM. The single-user scrollable panel has been replaced by a multi-user radar panel. The window on the right displays a miniature overview, or *radar view*, of the scrollable panel's content (mixed text and images, in this document). A uniquely colored and shaded rectangle indicates each participant's viewport location in the panel. Three participant viewports are shown in this example: *PC User* and *Solaris User* overlap near the top of the document while *Unix User* is near the center.

of this approach and the use of semantic knowledge to realize these advantages.

Flexible JAMM

We have incorporated our approach into a replicated-architecture collaboration-transparency system, called JAMM [4]. The new system is implemented with the Java Development Kit 1.1.5 and is called Flexible JAMM.

Figure 2 shows a single-user text editing application, called Stylepad, which is included as a sample application in the Java Foundation Classes graphical user interface library, called Swing [11]. This application contains a scrollable panel that in turn embeds a text editor. In Figure 3 the same application is now being shared among three participants via Flexible JAMM. The scrollable panel object has been replaced by a multi-user *radar* panel. Gutwin and Greenberg have demonstrated that multi-user radar views enhance the usability of collaborative applications for some collaborative tasks [10].

Flexible JAMM Target Platform We have constrained the set of applications that can be shared to those that are developed strictly using the Java Foundation Classes user interface library, Swing [11]. Flexible JAMM's application migration mechanism further limits its set of shareable applications. Flexible JAMM uses an **image-copy** scheme based on Java Object Serialization (JOS) to transfer the current state of a shared application to new-comers of the session. JOS is part of the core Java class library in Version 1.1 and provides the means to copy and transfer an object. For an application to be copied via JOS, however, the developer must choose to make all application classes **serializable**. Although in most cases serializability is not difficult to implement, a developer may have no need of it and therefore neglect to include that capability.

Collaboration-transparency systems generally allow the sharing of any application written for a particular platform (e.g., X or Windows). Due to the constraints outlined here, Flexible JAMM's goal is not to permit transparent sharing of the entire set of Java applications, but rather the subset of serializable, Swing-based Java 1.1 applications.

Dynamic Object Replacement In Java, polymorphism is provided such that an instance of a subclass may be referred to as its superclass. Therefore, an instance of a multi-user subclass may replace an instance of its standard single-user superclass.

In Flexible JAMM this replacement is provided by JOS. For each object reachable from the initial target object, JOS provides a mechanism to inspect and replace it with an instance of a polymorphically-equivalent class. Thus, the replacement of one object with another is transparent to the application.

Flexible JAMM replaces instances of two Swing classes at the time an application is shared. Each instance of `com.sun.java.swing.JScrollPane` is replaced by a multi-user radar pane, providing several benefits to collaboration: location-relaxed WYSIWIS, detailed group awareness (through the radar view), and immediate local response to scroll position changes. Flexible JAMM also substitutes each instance of `com.sun.java.swing.-text.PlainDocument`, which is the default data model of a Swing text component, with a multi-user document, providing significant groupware benefits: independent concurrent work, and immediate local response to edits. We selected these two classes for replacement, because we considered the development cost to be reasonable and the benefits to collaboration to be considerable.

As a replicated collaboration-transparency system, Flexible JAMM intercepts and distributes low-level input events. In addition, the Java 1.1 event delegation model allows Flexible JAMM to register itself as a listener to higher-level user interface actions, such as scrollbar position changes. This registration is transparent to the original single-user application. Thus, Flexible JAMM is notified of the semantic result of certain low-level input events. We next describe the use of the semantic knowledge of input events to the multi-user replacements to provide key groupware benefits: relaxed WYSIWIS, detailed awareness information, flexible floor control, concurrent work, and fast local application response.

Relaxed WYSIWIS Replication and semantic knowledge allow a flexible collaboration-transparency system to relax WYSIWIS by providing participants with independent views of the shared application. In contrast to the traditional replicated approach, in some cases our system does not distribute a given input event at all, but only applies it locally.

For example, the result of an input event within a scrollbar is to move the position of the scrollbar and in turn a viewport. In this case, a flexible collaboration-transparency system need not distribute the mouse event to all replicas, but only apply it locally, because each participant is allowed to have an independent viewport. Hence, the flexible collaboration-transparency system provides location-relaxed WYSIWIS.

Note that it is possible for an application to include the view position in its state. In that case, relaxed WYSIWIS permitsthe replicas to become inconsistent. A flexible collaboration-transparency system should allow the collaborators to turn off view relaxation, falling back to the traditional replicated approach of distributing all input events, thus preventing replica inconsistency but losing the advantage of relaxed WYSIWIS.

Awareness Information Allowing location-relaxed WYSIWIS means that replicas of a component may have independent state. To allow participants to coordinate their work, it is necessary to provide information to them about each others' view position.

Continuing the previous example, after the scroll position is changed locally, the system should send a message to remote participants indicating the new local viewport position. In response to a viewport change notification, a remote replica will update its location-awareness

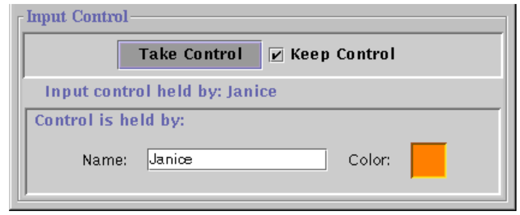

Figure 4: Flexible JAMM's input control panel indicating that control is currently held by the collaborator named Janice. Input control may be explicitly requested by pressing the "Take Control" button, or implicitly when a participant presses a mouse button in the shared application.

interface elements, such as the viewport indicators in the radar view used by Flexible JAMM (see Figure 3). Therefore, in addition to telepointers, Flexible JAMM provides group awareness information by indicating each participant's viewport position in the radar view.

Explicit and Implicit Floor Control To avoid potential conflicts between non-atomic events such as mouse drags [2], Flexible JAMM uses a floor control mechanism for some, but not all, interface components. Where floor control is used, Flexible JAMM provides mechanisms for both **explicit** floor requests, where participants explicitly grab and release control of the application, and **implicit** floor requests, where the system automatically makes floor requests as collaborators use the application. Implicit control provides seamless floor passing as work flows among the participants.

Figure 4 shows the user interface to Flexible JAMM's floor control mechanism. Control is implicitly granted to the first participant to begin a potentially non-atomic series of inputs (i.e., mouse drags) by pressing a mouse button. That participant holds control until he or she releases the mouse button and for a short time (5 seconds) after. The short delay to release the floor allows the same user to quickly regain control if attempted. Control may also be explicitly requested by pressing the "Take Control" button. If the current control holder checks the "Keep Control" check-box, no one else may gain control. Otherwise, control will be granted to the next participant who implicitly or explicitly requests it.

Concurrent Work Although implicit floor control facilitates control passing, it still allows only one participant at a time to work in the shared application. However, floor control locking is not required for inputs to all components. In the case of an input in a text-editing component (i.e., `com.sun.java.swing.JTextArea`), Flexible JAMM does not impose floor control. All participants may simultaneously edit the text.

To provide this concurrent editing capability, Flexible JAMM replaces the original text data object with an extension that uses serialized transactions to update replicas. For a key or mouse event generated in a text area, Flexible JAMM does not broadcast the event, but applies it locally only. If the document is changed as a result, which is usually the case, Flexible JAMM distributes a message describing the change to all replicas. The algorithm Flexible JAMM uses is an optimistic transaction approach similar to that described by Ellis and Gibbs, using no locking yet avoiding conflicts between edits that occur as close as one character apart [7]. A replica updates its copy of the document by transforming an incoming remote edit to be consistent with its own set of local edits.

Fast Local Response Flexible JAMM does not forward all user input events; some are applied only locally. As described in the previous subsection, key and mouse events that occur within a text area are not broadcast, but are only applied locally. Similarly, inputs that occur within a scrollbar of a radar pane are also only applied locally. In both cases, any resulting state change is sent some time after the input event. Thus, the local replica of the shared application responds immediately to the input and remote replicas are updated asynchronously.

EVALUATION

We have criticized conventional collaboration-transparency systems because they impose a tightly-coupled style of collaboration. However, when collaborators are working closely together, conventional collaboration-transparency systems should be adequate. Our approach to flexible collaboration transparency aims to provide equivalent support for tightly-coupled tasks and extend the conventional concept of collaboration transparency by additionally supporting loosely-coupled tasks. By supporting both ends of the range of collaboration styles, our flexible collaboration-transparency system hopes to allow a collaboration to flow naturally between the two.

Flexible JAMM provides features previously seen only in collaboration-aware applications: concurrent work, relaxed WYSIWIS and detailed group awareness. Each of these features has been shown to be useful in the context of an application designed to support collaborative work [7, 10, 15]. By introducing new interface elements (i.e., radar-views and concurrent editing) into a legacy single-user application, we were concerned that users' unfamiliarity with the new interface items would offset the advantages provided by our flexible system. We wanted to be sure that Flexible JAMM did not detract from the usefulness already present in conventional collaboration transparency. Therefore, we conducted a study to evaluate whether our flexible collaboration-transparency system did indeed provide better support for loosely-coupled collaborative tasks and did not adversely impact the performance of tightly-coupled collaborative tasks.

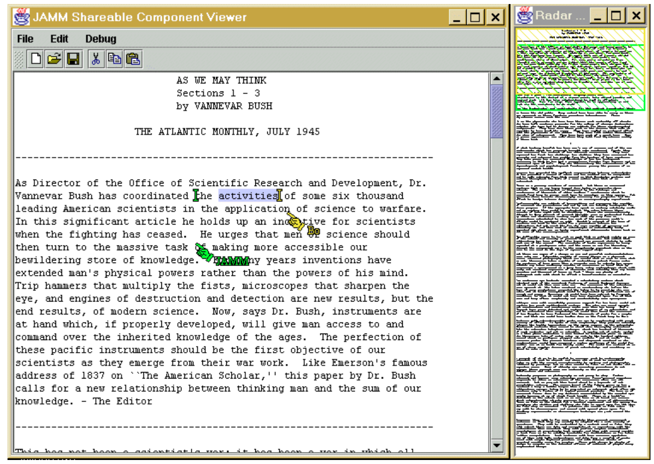

Figure 5: A text editor, Notepad, shared via Flexible JAMM.

Procedure and Methods

The study consists of two independent variables: 1) the type of collaboration-transparency system and 2) the type of task (loosely- versus tightly-coupled). In preparation for this experiment, we conducted a pilot study with four pairs of participants to refine the tasks and questionnaires as well as determine the number of participants needed.

Collaboration-transparency Software We were interested in comparing our flexible collaboration-transparency system, Flexible JAMM, versus a representative conventional collaboration-transparency system. Microsoft's NetMeeting was used as a representative conventional collaboration-transparency system, because it is freely available and is perhaps the most commonly used collaboration-transparency system today. Flexible JAMM and NetMeeting have different target platforms. Flexible JAMM targets serializable Swing-based Java applications, and NetMeeting targets Windows applications. However, since Java applications can be run on Windows, NetMeeting's target contains Flexible JAMM's target. For the evaluation, we used an application that can be shared by both, a Java-based text editor, called Notepad, which is shown in Figure 5 being shared via Flexible JAMM.

Tasks The tasks in this study involved editing text. We were interested in evaluating the effectiveness of the two collaborative software systems for tightly- versus loosely-coupled collaborations. Two types of task were used. The tightly-coupled task, called **Copy Edit,** was modeled after a scenario we have commonly experienced in which a manuscript editor guides an author to make changes to an article. This is a text-editing task analogous to the equipment-assembly task of Chapanis' communication study in which a **source** guides a **seeker** to construct a piece of equipment [5]. For Copy Edit, participants shared an electronic copy of a document that

Number	Order			
	1	2	3	4
1	FJ-C	FJ-T	NM-C	NM-T
2	FJ-C	FJ-T	NM-T	NM-C
3	FJ-T	FJ-C	NM-C	NM-T
4	FJ-T	FJ-C	NM-T	NM-C
5	NM-C	NM-T	FJ-C	FJ-T
6	NM-C	NM-T	FJ-T	FJ-C
7	NM-T	NM-C	FJ-C	FJ-T
8	NM-T	NM-C	FJ-T	FJ-C

Table 1: Counterbalanced ordering of task type nested in software type. Task types: C = Copy Edit, T = Text Entry. Software types: FJ = Flexible JAMM, NM = NetMeeting. Thus, NM-T means NetMeeting was used to perform the Text Entry task.

contained errors. One participant assumed the role of a manuscript editor and was given a hard copy of the seven-page document with ten corrections highlighted by ***bold-italic*** type. The other participant, in the role of the author, was the only one allowed to make the changes to the text. The manuscript editor was allowed to use the application in any way other than editing text.

The scenario used to evaluate the software in a loosely-coupled collaborative task modeled coauthors writing independently in an article. In this task, called **Text Entry,** the participants were given a two-paragraph article to enter into the editing software. The paragraphs had the same number of words (107) and approximately the same number of characters (601-611). The participants were instructed to each enter one paragraph and decided between themselves who entered which. For both tasks, participants were instructed to "work as quickly and accurately as you can."

Design For this study, we used a two-factor factorial in a randomized complete block design. The task type was one factor and the collaboration-transparency system was the other. Each pair of participants was considered a block and received all four treatment combinations. This within-subjects design allows participants to make comparisons when they rate the software and helps offset the notoriously high variability of groups, because it assumes that each block (pair of participants) is heterogeneous. The order of the tasks was nested within the order of the software in a counterbalanced design shown in Table 1. Each order combination was randomly assigned to one group.

Participants Thirteen male and three female computer science undergraduate students with a minimum typing proficiency of thirty-five words-per-minute participated in the study. The sixteen participants were each paid \$7.50 US. Participants were allowed to choose a partner or were paired with another by the experimenter. Although some of the participants had heard of NetMeeting, none had previously used Flexible JAMM or NetMeeting for application sharing.

Procedure Each session took approximately one hour. Participants signed a consent form and were given a copy of it. They each then filled out a short demographic information form and took a short typing speed test. Next, they were shown an approximately one and a half minute long instruction video, which demonstrated using the first collaboration-transparency system to share the text editor application. The instruction include the features of the collaboration-transparency system that would be needed to collaborate on the tasks. For Flexible JAMM, the video defined telepointers, showed how to use the radar view, and demonstrated concurrent text entry. For NetMeeting, the video defined floor control, explained how to pass control, and demonstrated passing control between two users. Participants then practiced using the text editor shared via the collaboration-transparency system until they felt comfortable with it (typically, less than one minute). They then used the software to perform the Copy Edit and Text Entry tasks, filling out the same questionnaire after each.

Before being introduced to the next system, participants rested for five minutes during which a computer-animated cartoon was shown. Then the same procedure was followed for the second collaboration-transparency system. Following that, the participants filled out a questionnaire that asked which system they preferred overall and for each type of task. Finally, the experimenter briefly interviewed the participants about observed critical usability incidents, their impressions about both systems, and alternative interface details for Flexible JAMM.

Equipment and Layout Because there is more than one participant and computer involved, the typical laboratory setup for the evaluation of single-user software is inadequate to capture a collaborative session. Fortunately, we found that equipment commonly used for single-user software studies is sufficient, though not ideal, for collaborative studies.

The layout of the experiment site is shown in Figure 6. Participants were seated across from each other; the monitors are partially sunk into the desktop allowing participants' a clear view of each other's face. The audio of each participant was recorded. Two video signals were recorded. First, the screen content of one participant was captured with a scan converter. Although in one condition each participant can have an independent view of the shared application, we found capturing just one screen was adequate. Second, a video camera was placed behind the other participant, which encompassed

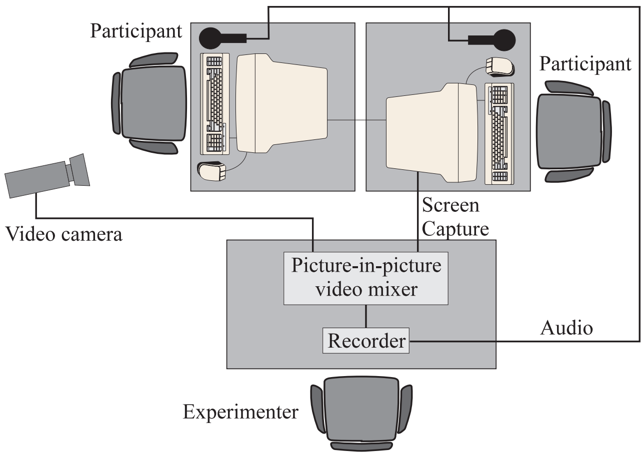

Figure 6: The layout of the experiment site.

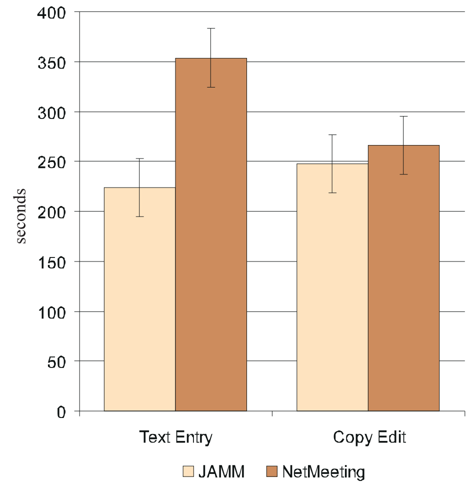

Figure 7: Least squares mean seconds to complete task. Error bars indicate 95% confidence interval of actual mean.

a view of this participant's screen and both participants. The two signals were mixed and recorded such that the screen capture was full size and the camera view was one-quarter size (picture-in-picture). The camera view was dynamically moved and resized so as to not obstruct the activity in the software. The screen capture and audio are the primary media needed to review a session. Generally, the camera view was only detailed enough to distinguish body gestures made by participants and was primarily useful only to complete the record.

	Task	
	Text Entry	Copy Edit
Flexible JAMM	223.75	247.50
NetMeeting	353.50	266.00
p	**0.0001**	0.7905
95% confidence interval $= \bar{x} \pm 29.33$		

Table 2: Least squares mean seconds to complete task. Significant difference is indicated by **bold** values for p.

Results

The focus of the evaluation is the interaction of task type and collaboration-transparency system. We used a two-way analysis of variance (ANOVA) to compare completion times, accuracy, and perceived effort. We used an alpha of 0.05 to determine significance.

Completion Time The interaction of task and system is significant for completion times ($F = 15.56$, $p = 0.0007$). The results show that participants completed the Text Entry significantly faster using Flexible JAMM than using NetMeeting. The results are summarized in Figure 7 and Table 2. To determine the amount of difference, we conducted pre-planned comparisons of the least squares means (Tukey adjusted for multiple comparisons) of each combination of system and task. For Text Edit, the difference in completion time using Flexible JAMM versus NetMeeting is between 74.15 and 185.34 seconds at 95% confidence. For Copy Edit, there is no significant difference between the two systems.

For the most part, these results agree with our expectations. For the Copy Edit task, we expected completion times to not be substantially dependent on the system, but perhaps slightly longer using Flexible JAMM because each participant has to scroll through the text independently, as opposed to NetMeeting where both scroll simultaneously. In fact, the observed mean completion times for the Copy Edit task were insignificantly faster for Flexible JAMM. To confirm that the observed significant results were not unduly influenced by transfer effects due to ordering, we also performed a between-subjects analysis using only the data from the first system used by each group. The results of the between-subjects analysis correspond to those of the within-subjects analysis with $p = 0.0001$ for Text Entry and $p = 0.364$ for Copy Edit.

Accuracy We counted the number of errors found in the resulting text after the completion of each task. For this measure, there was no significant interaction of task and system ($F = 1.11$, $p = 0.3051$). The number of errors is the same regardless of the system used. The results are summarized in Table 3.

For Copy Edit, the results matched our expectation. However, for the Text Entry task we expected the accuracy to be better using NetMeeting, because only one participant can enter text at a time allowing the other to watch for errors. However, although the observed mean number of errors was less with NetMeeting, the difference was not significant. The lack of substantial difference is most likely because only one pair thought of and used the error-checking strategy while using NetMeeting. Again, the results of the between-subjects analysis correspond to those of the within-subjects analysis with

	Task	
	Text Entry	Copy Edit
Flexible JAMM	2.25	0.50
NetMeeting	1.25	0.50
p	0.1519	1.0000
95% confidence interval = $\bar{x} \pm 0.989$		

Table 3: Least squares mean number of errors per task and system. No significant difference was observed.

$p = 0.280$ for Text Entry and $p = 0.659$ for Copy Edit.

User Perception Following each task, participants were asked a set of questions to measure their opinion about aspects of the collaboration-transparency system's support of the collaboration for that task. The questions were formed as statements to which the participants marked the degree to which they disagreed or agreed (strongly, moderately, mildly) or neither. The seven questions asked participants their perception of 1) satisfaction with the collaborative software, 2) ability to work simultaneously, 3) ease of controlling the shared application, 4) ease of indicating text locations to partner, 5) ease of simultaneously editing text, 6) ability to have independent views, and 7) ease of knowing partner's view position.

The responses for the Text Entry task are summarized in Figure 8. For all questions, the responses were significantly positive using Flexible JAMM, whereas only question 3 received a significantly positive response using NetMeeting. There are significant differences in the responses between the two systems for all but questions 3 and 4 ($p < 0.001$). For this task, these responses generally correspond to the observed use of each collaboration-transparency system in the performance of the task. The lack of distinction in question 4 is because participants generally did not indicate text locations to each other, although we thought they would to point out errors. The significant difference in response to question 7 is unexpected. Since users share the same view in NetMeeting, users should implicitly know their partner's view position. The observed difference in the response then is possibly due to the implicit nature of this knowledge using NetMeeting whereas with Flexible JAMM users were made aware that view position information was relevant to the completion of the task.

The responses for the Copy Edit task are summarized in Figure 9. Participants responded significantly positively for all questions using Flexible JAMM and for questions 4 and 7 using NetMeeting. There are significant differences in the responses between the two systems for questions 1 ($p = 0.0132$), 2, 3, 5, and 6 ($p < 0.003$). We note that the unexpected difference in response to question 7 for the Text Entry task, described previously, does not show for the Copy Edit task.

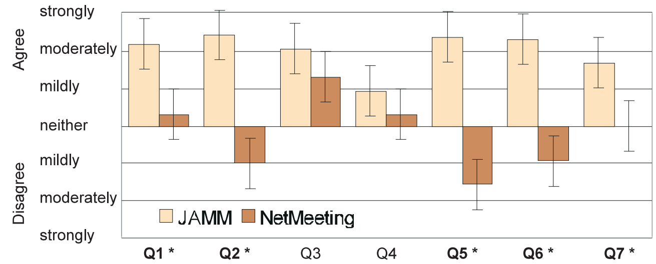

Figure 8: Mean responses to questions after the Text Entry task. Error bars indicate 95% confidence interval of the actual mean. Asterisks indicate significant difference in the responses for that question for Flexible JAMM versus NetMeeting.

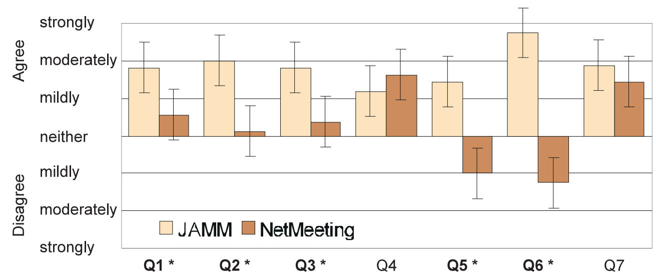

Figure 9: Mean responses to questions after the Copy Edit task. Error bars indicate 95% confidence interval of the actual mean. Asterisks indicate significant difference in the responses for that question for Flexible JAMM versus NetMeeting.

Unlike the objective measures of completion time and number of errors, the results of the between-subjects analysis for post-task question response did not exactly correspond to the within-subjects results. For the Text Entry task, significant difference was seen for questions 2, 5, 6 ($p < 0.005$) and 7 ($p = 0.0142$), whereas only question 6 showed significant difference ($p = 0.0001$) for the Copy Edit task. The lack of distinction in response to the Copy Edit task is likely due to the participants not having a base from which to make judgments. This was observed in questionnaire responses in a study by Gutwin and Greenberg [10] and was a primary reason we chose to use a within-subjects design for this study.

Overall Preference

At the completion of all tasks using both systems, participants were asked which system they preferred overall as well as which system better supported collaboration overall, better supported the Copy Edit task, and better supported the Text Entry task. Respondents overwhelmingly selected Flexible JAMM ($\chi^2 = 16$, $p < 0.001$) for all except for support of Copy Edit, where neither system was selected over the other ($\chi^2 = 0.29$, $p = 0.9$). Table 4 displays the results.

Question	Flexible JAMM	NetMeeting
Prefer overall	16	0
better supported:		
collaboration	16	0
Text Entry	16	0
Copy Edit	8	6

Table 4: Preference scores for each system.

Results Summary and Discussion

Analysis of the measured results confirmed our expectation in terms of task completion time: participants were able to complete the Text Entry task more quickly using Flexible JAMM than using NetMeeting. Participants used the same amount of time to complete the Copy Edit task using each system. The number of errors was not significantly different for either task using either system. Users rated Flexible JAMM significantly positively for both tasks in each of a series of seven questions for both tasks, whereas NetMeeting was rated significantly positively for only one question in the Text Entry task and two in the Copy Edit task. The responses were significantly different between Flexible JAMM and NetMeeting in five of the seven questions for both tasks.

The faster completion time in the Text Edit task is primarily because Flexible JAMM allows simultaneous text entry whereas NetMeeting does not. However we also saw somewhat faster times for Flexible JAMM in the Copy Edit task, where simultaneous text entry is not a factor. Part of this can be explained by the awkwardness of passing control of the application in NetMeeting, whereas it is not necessary to pass control when entering text in Flexible JAMM. Participants rated NetMeeting significantly lower than Flexible JAMM in response to question 3 (ease of controlling the shared application) for the Copy Edit task (see Figure 9). To gain control in NetMeeting, the requester needs to click the mouse or press a key once. When a key press is used, the character pressed is consumed, which many times was a character that a participant intended to enter in the text. Moreover, users commented that when they had to take turns in NetMeeting, they did not feel like they were working together. Thus even though passing control is fairly simple, interface difficulties and turn taking diminished the benefits of collaboration. In contrast, participants commented that they felt more involved using Flexible JAMM, because no floor control was required.

CONCLUSIONS

Conventional implementations of collaboration transparency impose an inflexible style of collaboration and do not adequately support key groupware principles: concurrent work, relaxed WYSIWIS, detailed group awareness, and efficient network usage. We have presented an approach to flexible collaboration transparency that supports these groupware principles, allowing a collaboration to shift between loosely- and tightly-coupled work. Our approach is based on an object-oriented replicated architecture where selected single-user interface objects are dynamically and transparently replaced by multi-user extensions. As an example of this approach, we described its incorporation into a Java-based collaboration-transparency system, called Flexible JAMM.

We conducted a study to evaluate whether our flexible collaboration-transparency system did indeed provide better support for loosely-coupled collaborative tasks without adversely impacting the performance of tightly-coupled tasks. Participants were able to complete a loosely-coupled task significantly more quickly with the same accuracy when using Flexible JAMM than when using NetMeeting. No significant difference in completion time or accuracy was observed between the two systems when participants performed a tightly-coupled task. Participants preferred Flexible JAMM overall and commented that they felt involved in the collaboration, whereas with NetMeeting, the explicit turn taking made them feel like they were not working together. The results support the aim of our system to provide flexible support of multiple styles of collaboration in a collaboration-transparency system.

We have found that it is possible to bring collaboration transparency closer to the advantages afforded by collaboration awareness. By supporting the range of tightly- to loosely-coupled collaboration styles, our flexible collaboration-transparency system allows a collaboration to flow naturally between the two. This capability enhances the collaborative use of the growing body of legacy single-user applications.

Flexible JAMM is available at http://simon.cs.vt.edu/-jamm.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of the National Science Foundation under grant DGE-9553458. We also appreciate the assistance of Dennis C. Neale and Ray A. Reaux in the questionnaire design.

REFERENCES

1. H. Abdel-Wahab and M. Feit. XTV: A Framework for Sharing X Window Clients in Remote Synchronous Collaboration. In *Proceedings of IEEE TriComm*, pages 159–167, 1991.

2. J. B. Begole and C. A. Shaffer. Flexible Collaboration Transparency. Technical Report TR-98-11, Department of Computer Science, Virginia Tech, Blacksburg, VA, Apr. 1998, submitted to The 1998 ACM Conference on Computer Supported Cooperative Work.

3. J. B. Begole, C. A. Struble, and C. A. Shaffer. Leveraging Java Applets: Toward Collaboration Transparency in Java. *IEEE Internet Computing*, 1(2):57–64, Mar./Apr. 1997.

4. J. B. Begole, C. A. Struble, C. A. Shaffer, and R. B. Smith. Transparent Sharing of Java Applets: A Replicated Approach. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST-97)*, pages 55–64, New York, Oct.14–17 1997. ACM Press.

5. A. Chapanis. Interactive human communication. *Scientific American*, 232(3):36–42, Mar. 1975.

6. P. Dewan and R. Choudhary. Flexible User Interface Coupling in a Collaborative System. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, Group Use of Computing, pages 41–48, 1991.

7. C. A. Ellis and S. J. Gibbs. Concurrency Control in Group Systems. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 18(2):399–407, June 1989.

8. D. Garfinkel, B. Welti, and T. Yip. HP SharedX: A Tool for Real-Time Collaboration. *HP Journal*, pages 23–26, Apr. 1994.

9. S. Greenberg and M. Roseman. GroupWeb: A Groupware Web Browser. In *Proceedings of the ACM 1996 Conference on Computer Supported Work, Video Program*, page 7, New York, Nov.16–20 1996. ACM Press.

10. C. Gutwin and S. Greenberg. Effects of Awareness Support on Groupware Usability. In *Proceedings of the 1998 ACM SIGCHI Conference on Human Factors in Computer Systems*, New York, Apr.18–23 1998. ACM Press.

11. Java Foundation Classes, Mar. 23, 1998. <http://-java.sun.com/products/jfc>.

12. J. C. Lauwers, T. A. Joseph, K. A. Lantz, and A. L. Romanow. Replicated Architectures for Shared Window Systems: A Critique. In *OIS90*, Computer Mediated Work Environments, pages 249–260. ACM Press, 1990.

13. Microsoft NetMeeting, Mar. 23, 1998. <http://-www.microsoft.com/netmeeting/>.

14. G. Smith. Cooperative Virtual Environments: Lessons from 2D Multi User Interfaces. In *Proceedings of the ACM 1996 Conference on Computer Supported Work*, pages 390–398, New York, Nov.16–20 1996. ACM Press.

15. M. Stefik, D. G. Bobrow, G. Foster, S. Lanning, and D. Tatar. WYSIWIS Revised: Early Experiences with Multiuser Interfaces. *ACM Transactions on Office Information Systems*, 5(2):147–167, 1987. An earlier version appeared in CSCW '86.

16. ShowMe SharedApp, Mar. 23, 1998. <http://-www.sun.com/products-n-solutions/sw/-ShowMe/products/ShowMe_SharedApp.html>.