Collaboration Bus: An Infrastructure for Supporting Interoperating Collaborative Systems

Prasun Dewan
Kevin Jeffay
Lars Nyland
John Smith
Dave Stotts
University of North Carolina at Chapel Hill

Hussein Abdel-Wahab
Old Dominion University

John McHugh
Portland State University

John Menges
Hewlett Packard

PROPOSAL ABSTRACT

Section I. Administrative

A. Cover

BAA No: 96-06

Proposal Abstract Control No: 9606042

Technical Topic Areas: Collaboration Infrastructure and Systems Integration Technology

Proposal Title: Collaboration Bus: An Infrastructure for Supporting Interoperating Collaborative Systems

Technical Point of Contact:

Prasun Dewan
919 962 1823 (work)
dewan@cs.unc.edu
919 962 1799 (fax)
University of North Carolina
Department of Computer Science
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175

Administrative Point of Contact:

Tim Quigg
919 962 1777 (work)
quigg@cs.unc.edu
919 962 1799 (fax)
University of North Carolina
Department of Computer Science
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175

Contractor's type of business: OTHER EDUCATIONAL (Graduate Research and Teaching)

Section II. Summary of Proposal

A. Innovative Claims

In this project, we will address the problem of composing new collaborative systems from existing single-user and collaborative systems. We propose to develop a new software abstraction, called the *collaboration bus*, that makes it easy to compose collaborative systems. The collaboration bus will be an extensible infrastructure that provides general definitions of collaboration services, default implementations of these services, rules for interconnecting these services, and mechanisms for extending the set of supported services.

The set of generic services supported by the bus will include (a) a data model for defining shared workspaces consisting of structured objects; (b) a user interface model that allows shared objects to be manipulated using user interfaces that support text, graphics, audio, video, and virtual reality; (c) a coupling model that enables users in a joint session to share data and user-interface objects and controls their interactions to ensure security and consistency contraints are not violated; (d) migration and replication mechanisms that ensure that the bus does not become a bottleneck; and (e) real-time services, consisting of scheduling algorithms and network protocols, for ensuring real-time coupling among the collaborators. Previous work has addressed the heterogeneity problem in other domains such as software engineering, operating systems, and database systems. Our project will extend this research by providing an interconnection technology for the collaboration domain. It will offer two forms of interoperability: (1) inter-service interoperability, which will allow existing implementations of *different* collaboration services, such as user-interface and real-time services, to be combined in new collaborative systems; and (2) intra-service interoperability, which will resolve inconsistencies among different, competing instances of the *same* service, such as explicit and implicit schemes for creating joint sessions, provided by different existing systems that need to interoperate with each other.

We will not only define and implement the collaboration bus, but also evaluate its efficiency, programmability, and flexibility by performing interconnection experiments among a group of existing single-user and collaborative software systems. These experiments will demonstrate inter-service interoperability among at least three different services and intra-service interoperability among at least three different instances of the same service. In particular, we will demonstrate inter-service interoperability among object-oriented Java data model, 3-D VE user interfaces build at UNC, real-time support developed at UNC, and coupling services; and intra-service interoperability among different forms of coupling implemented in state-of-the-art desktop conferencing systems offered by HP, UNC, and ODU. We have previously investigated direct interoperation between some of these systems; and used general-purpose interconnection technology to compose collaborative virtual environments. Our previous integration experiments will give us baseline data for determining the improvements obtained with the bus.

B. Deliverables

The main deliverables of this project are:

- *General definitions and design space*: General definitions and design spaces of data, user-interface, coupling, and real-time model.
- *Bus Infrastructure*: A replicated, mobile, and secure infrastructure that interconnects existing implementations of these services and provides default implementations of these services.
- *Demonstrational Interconnections*: A set of experimental interconnections among existing, state-of-the-art, single-user and collaborative systems that demonstrate inter-service interoperability among at least three different services and intra-service interoperability among at least three different implementations of the same service. In particular, we will demonstrate inter-service interoperability among object-oriented Java data model, 3-D VE user interfaces build at UNC, real-time support developed at UNC, and coupling services; and intra-service interoperability among different forms of coupling implemented in state-of-the-art desktop conferencing systems offered by HP, UNC, and ODU.
- *Reports and Videos*: Technical reports and videos that illustrate and describe the capabilities of the infrastructure and demonstrational interconnections and report the results of our interconnection experiments.

There are no proprietary claims.

C. Cost, Schedule, and Milestones

The milestones and the approximate associated costs are summarized below. The table shows when different collaboration services will be incorporated into the bus, when experiments will be carried out to demonstrate the interoperability among software systems providing these services, and the expected cost of carrying out these tasks.

We will incorporate and experiment with these services in three stages. In the first stage, we will support *application-independent* aspects of our work such as session management and awareness support and conduct experiments that allow interoperability among different implementations of these services. Next, we will support application-dependent services for *enabling* real-time collaboration among interoperating systems. These services include support for a general data and user-interface model and mechanisms for real-time coupling of application-dependent data and user-interface objects. We will then investigate application-dependent services for *controlling* the collaboration among users to ensure that unauthorized and inconsistent operations are not performed on the interoperating applications. Finally, we will study replication and migration mechanisms that ensure that connection through the bus does not unduly affect the performance of the interconnecting processes and access control mechanisms for ensuring that operations on the the bus itself are also protected.

The equipment cost shown below includes the cost of two SGI Indigo2 Maximum Impact workstations ($40,000 cost per unit), which will facilitate the study of collaborative virtual environments.

Milestone	**Year 1**	**Year 2**	**Year 3**	**Total**
Acquire and install new equipment	40,000	40,000		80,000
Identification of Initial Bus Clients (Smith) and Integration Scenarios	50,000			
Session management and awareness (Wahab)	50,000	50,000	50,000	
Data and UI model (Stotts)	50,000	50,000	50,000	
Real-Time Services (Jeffay)	50,000	50,000	50,000	
Sharing mechanisms (Dewan)	50,000	50,000		
Concurrency and access control services (Smith)		50,000	50,000	
Bus replication/migration mechanisms (Dewan)			50,000	
Bus security (McHugh)	50,000	50,000	50,000	
Total	$340,000	$340,000	$300,000	$980,000

Each milestone corresponds to a function we propose to add in the bus. Included in the work associated with the milestone is the cost of implementing the function, interfacing clients/providers of that service to the bus, performing experiments involving the function, and producing reports and videos involving this work. Next to each milestone we have given the name of the faculty member who will lead the research of that function.

D. Technical Rationale and Approach

Rationale

As a motivation for the concept of a collaboration bus, consider two conferencing scenarios:

(1) *Scenario 1*: Suppose four geographically separated users, A, B, C, and D, wish to collaboratively discuss and annotate a mission plan document. Each one of them is using a text editing facility to compose notes and annotations for the document. Collaborators A and B use a specialized multiuser editor that allows them to couple their text buffers without coupling their scrollbars, lock fine-grained objects such as paragraphs, and link to nodes in the document being examined. Users C and D are unwilling to learn this editor, and prefer instead to use shared window systems, which allow them to share existing single-user editors. Each of them, however, prefers a different shared window system. User C wishes to use ODU's XTV shared window system while user C wishes to use HP's Shared X. Both shared window systems provide capabilities to completely couple windows of different users, but differ in the window system architecture, concurrency control, and session management they support.

(2) *Scenario 2*: Consider now an architectural VE built for visualizing new ship designs; it allows an individual to construct a 3-D model of a new ship and to "walk through" the ship, experiencing the design from a ship-dweller's perspective. Consider also a second VE built to simulate a group conference; it allows a collection of individuals physically remote from one another to experience a face-to-face meeting in a virtual 3-D environment. Could we not combine these two systems so they interoperate, allowing a group of physically separated people to virtually walk through a ship "together," communicating as a group?

The services offered by a variety of single-user and collaborative systems are being used above to facilitate two distributed, real-time collaborations. Without this integration, the collaborations may not have taken place. In the first case, the different users may not have been able to agree to use a specific window system and text editor. In general, formation of work groups is dynamic, usually in response to an identified need, and brings together people with different training, different skills, and different work habits. Therefore it is crucial to support scenarios such as the first one. In the second case, the alternative to integration of the two systems is a new system that supports the capabilities of these systems. However, the expense and time required to build new collaborative systems makes the integration approach far more attractive.

These scenarios are currently far from possible since current collaboration tools have not been designed to work with other tools and several complex issues must be resolved before these tools can be made to interoperate with each other. This problem is not peculiar to collaboration systems and specific solutions have been addressed for heterogeneous programming languages, operating systems, database systems, and software engineering. This project will explore the instance of this problem that has arisen in the area of collaboration. It is our hypothesis that it is possible to develop a new abstraction, called the collaboration bus, that interconnects existing single-user applications and collaboration systems and facilitates new, useful kinds of collaborations at far less expense than existing abstractions. This abstraction, like interconnecting technologies in other domains, can be expected to be an effective solution to the heterogeneity problem since connecting to a bus abstraction is far cheaper than writing a cross-translation module for every different pair of tools that might want to interoperate.

Approach

The design of such a bus requires a general abstraction that covers the functionality of existing collaborative systems. In particular, the bus must capture similar solutions to a problem so that these solutions can be made to work across tools. For instance, it must capture the notions of (coarse-grained) floor control and fine-grained concurrency control so that inter-tool concurrency control control can be supported for all tools implementing these concepts. In addition, it must provide, when possible, semantic translation to allow dissimilar solutions to the same problem to coexist. For instance, it must be able to find a compromise when interfacing floor control and fine-grained concurrency control. It must also provide a definition of a collaboration service that is general enough to allow it to be composed with other services. For instance, it provide a definition of real-time services that can be applied to not only video conferencing but also shared virtual environments. Finally, the bus must not become a bottleneck that significantly increases the response times of users.

We intend to use the following approach in designing, implementing and evaluating the bus:

- Identify the services that a collaborative application requires.
- Identify a general definition and design space for each of these services. The definition should allow a service to be composed with all other services, thereby supporting inter-service interoperability. The design space will identify different methods for providing the service.
- Identify semantics translations among the different points in the design space of a particular service, thereby supporting intra-service interoperability.
- Incorporate, into the bus, a default implementation of the service and the associated semantic translations.
- Develop mechanisms and policies for replicating and migrating parts of it.
- Develop mechanisms for protecting the operations on the bus.
- Carry out interoperability experiments involving each of these services using state-of-the-art existing software.

A large number of services must be provided to support distributed collaboration. These services can be divided into application independent and dependent services based on whether they depend on the specific data structures and semantics of an application. An example of an application-independent service is session management, which groups together users wishing to use a set of applications, and does not depend on the insides of these applications. An example of an application-dependent service is access control, which controls access to objects of an application, whose types are defined by the application. Collaboration services can be further divided into collaboration enabling and controlling services, based on whether they enable communication among the collaborators or control the actions of the collaborators. An example of a collaboration-enabling service is sharing, which links objects belonging to different users. Access and concurrency control are examples of collaboration-control services. In this project, we will investigate both application dependent and independent services, and collaboration enabling and control services. Support of application-dependent services requires a model of the data and user interface of a collaborative application. Therefore, our collaboration bus will also include data and user-interface modelling components.

To be more specific, our bus will include: (a) a data model for defining shared workspaces consisting of structured objects; (b) a user interface model that allows shared objects to be manipulated using user interfaces that support text, graphics, audio, video, and virtual reality; (c) a coupling model that enables users in a joint session to share data and user-interface objects and controls their interactions to ensure security and consistency contraints are not violated; (d) migration and replication mechanisms that ensure that the bus does not become a bottleneck; and (e) real-time services, consisting of scheduling algorithms and network protocols, for ensuring real-time coupling among the collaborators.

To evaluate our research and demonstrate its effectiveness, we will use the bus to integrate a subset of the current collaboration and single-user systems offering a variety of services. We currently have numerous connections with collaborators at other sites, including IBM, Intel Corp., H.P., Xerox, Univ. of Florida, Univ. of Minnesota, Texas A&M Univ., Univ. of Michigan, University of Illinois, and Univ. of Adelaide (Australia). We expect to conduct our experiments in conjunction with these colleagues and the tools with which they customarily work. Thus, the subset of systems we have initially identified to integrate includes XTV (ODU), Suite (UNC), Shared X (HP), Trellis (UNC), ABC (UNC), DistEdit (Michigan), Web browsers, Java objects, and VE applications and real-time services developed at UNC. Integration of some of these systems is already being investigated on a case-by-case basis (such as Suite and Web, Trellis and Web, DistEdit and Suite, XTV and Suite, and Trellis and Suite), and should fit nicely into the proposed unified framework of the bus. Our efforts with individual integrations will give us baseline data for determining improvements obtained with the bus.

Our experiments will demonstrate inter-service interoperability among at least three different services and intra-service interoperability among at least three different implementations of the same service. In particular, we will demonstrate inter-service interoperability among object-oriented Java data model, 3-D VE user interfaces build at UNC, real-time support developed at UNC, and coupling services; and intra-service interoperability among different forms of coupling implemented in state-of-the-art desktop conferencing systems offered by HP, UNC, and ODU.

To illustrate the nature of the proposed research in intra-service interoperability, consider the session management service. Some session management schemes require users to explicitly join sessions while others implicitly include them in sessions based on the activities they performing. Some explicit schemes require new users to be invited to a conference while others allow users to autonomously join sessions they know about. Some implicit schemes join users in a session based on the artifact they are manipulating, while others join them based on their proximity in a virtual space. We will investigate techniques for interoperating applications offering different session management schemes, thereby allowing, for instance, one collaborator to use explicit session management and another to use VE-based implicit session management. Similarly, we will devise conflict resolution rules for handling inconsistencies in other services such as awareness and sharing.

To illustrate the nature of the proposed inter-service interoperability, consider the user-interface and real-time services. So far, real-time services have been studied mainly in the context of continuous video and audio. By investigating interoperability between real-time services and VE-based user interfaces, we will go beyond the problem of supporting ‘‘talking heads in a window’’ to consider shared distributed virtual worlds that allow users to collaborate with each other in real-time. Similarly, we will consider interoperability between other pairs of services such as access control and multimedia, concurrency control and virtual environments, and so on.

In general, integrating two existing systems will require that each be made "bus aware", that is, be modified to communicate with the collaboration bus. However, often, it is not desirable or possible to change a software system. Therefore, we intend to explore methods for interfacing with the bus that do not require changes to existing systems and explore the kind of integration methods that can be supported by these techniques. In our research on shared window systems, we have had substantial experience with the notion of writing filter-based "pseudo server", that is, a process placed between a client and a real server that pretends to be the real server and processes requests from the client before forwarding them to the real server. We intend to explore the use of this technique for interfacing unchanged systems to the bus.

We will incorporate and experiment with the collaboration services in several stages. In the first stage, we will support application-independent services and conduct (inter-service and intra-service) interoperability experiments that involve these services. Next, we will incorporate a data and user interface model for supporting application-dependent services. We will then research the application-dependent collaboration-enabling services, and finally, we will investigate application-dependent collaboration-control services. We will investigate these services in a bottom-up, breadth-first manner, including as much support as is needed to perform the current interoperability experiment, and gradually increasing the scope of our interoperability experiments.

Previous research efforts have addressed generic data models (e.g. CORBA) and multimodal user interfaces (e.g. MASTERMIND [67]). We will extend this research in three main ways. First, we will evaluate the suitability of existing, general purpose concepts for capturing the data and user-interfaces in our target collaborative applications and make any necessary adaptations. Second, we will add to the general-purpose models common objects found in the collaboration/multimedia domain. Finally, we will define interfaces between the data and user-interface models and other components of our collaboration model.

E. Previous Research

Our proposed work is related to and will build on a variety of previous research efforts. We outline below the nature of some of the related works and how we will extend them:

- *Specific collaboration services:* Several research projects, both within and outside our group, have researched the individual collaboration services we will support in the bus. However, these efforts have explored these services within limited domains. For instance, Prof. Jeffay's research on on real-time computing has been limited to audio and video, and Prof. Dewan's research on flexible coupling has been limited to textual user-interfaces. In this project, we will generalize the domain of the individual services so that arbitrary services can be combined with each other, thereby exploring, for instance, real-time, coupling, services for supporting shared virtual environments.
- *Direct integration experiments*: Our group has recently conducted several experiments to explore direct interoperation between specific collaboration systems. In particular, we have integrated Suite, a collaboration toolkit developed by Prof. Dewan at Purdue and UNC, with (a) XTV, a shared window system developed by Prof. Wahab and Jeffay at UNC and ODU, (b) Trellis, a petri-net based collaboration system developed by Prof. Stotts at University of Maryland and UNC, (c) DistEdit, a collaborative text editor developed by Prof. Prakash at University of Michigan, and (d) HTML-based Web browsers. We will expand this research by providing indirect connection, via the bus, among interoperating systems, thereby requiring a single connection from an existing system to the bus rather than a a cross-translation module for every different pair of systems that might want to interoperate.
- *Generic interconnection technology*: Prof. Stotts is exploring the use of the Polylith software bus for composing collaborative virtual environments from existing single-user virtual environments. Polylith in an example of a generic technology for interconnecting arbitrary distributed applications. Our work will extend such technology with higher-level abstractions that make it easier to interconnect those distributed applications that support collaboration.

Our previous integration experiments will give us baseline data for determining improvements obtained with the bus.

F. Key Personnel

The key personnel include six faculty members: Assoc. Prof. Prasun Dewan, Assoc. Prof. Kevin Jeffay, Prof. John Smith, Assoc. Prof. Dave Stotts, Prof. Hussein Abdel-Wahab, and Assoc. Prof. John McHugh. All faculty members are currently involved in several individual and group projects studying issues related to collaboration infrastructures. These projects are sponsored by NSF, NSA, ARPA, ONR, NIST, Intel, and IBM.

We will work with graduate research assistants, at UNC-Chapel Hill, ODU, and PSU. In addition, we will collaborate with John Menges of Hewlett Packard on both the definition of the bus and how the desktop conferencing products of the company should be adapted to interface to the bus. We have ongoing collaborations with our industrial partner. HP will to cover the cost of the efforts put in by John Menges and we have not included this cost in our budget.

We give below the amount of effort (in full time months) to be expended by each of the key personnel during each contract year.

Personnel	Year 1	Year 2	Year 3	Total
Dewan (UNC)	1	1	1	3
Jeffay (UNC)	1	1	1	3
Smith (UNC)	1	1	1	3
Stotts (UNC)	1	1	1	3
Abdel-Wahab (ODU)	1	1	1	3
McHugh (PSU)	1	1	1	3
Menges (HP)	1	1	1	3
GRA1	7.5	7.5	7.5	22.5
GRA2	7.5	7.5	7.5	22.5
GRA3	7.5	7.5	7.5	22.5
GRA4	7.5	7.5	7.5	22.5
GRA5	7.5	7.5	7.5	22.5
GRA6	7.5	7.5	7.5	22.5

The 7.5 month of student effort each year includes 9 months (academic year) at half time and 3 months (summer) at full time.

Section III. Detailed Proposal Information

A. Statement of Work

In this project, we will address the problem of composing new collaborative systems from existing single-user and collaborative systems. We propose to develop a new software abstraction, called the collaboration bus, that makes it easy to compose collaborative systems. The collaboration bus will be an extensible infrastructure that provides general definitions of collaboration services, default implementations of these services, rules for interconnecting these services, and mechanisms for extending the set of supported services.

We will perform the following specific tasks to meet this goal. Many of these tasks will be carried out in parallel by different members of our team.

(1) *Choose Data Model:* While we do not propose to develop new techniques for allowing interoperating agents to share a common data format, our bus must be based on a data model. We currently propose to use the Java data model. While Java does not currently address interoperability, there is work underway to make it compliant to the CORBA object-based interoperability framework. Moreover, we expect the techniques we develop to be easily implemented into any interoperable object-based model. This task will be complete once we have a data model powerful enough to define all of the collaboration types we design.

(2) *Develop User-Interface Model:* The Java programming environment provides a model for attaching user-interface abstractions such as scrollbars, buttons, forms, text editors, and sliders to Java objects. We will extend this model to attach audio, video, and VE-based user interfaces to semantic objects, and allow interoperability between different user interfaces to the same semantic object. This work will be complete when we can accommodate and interoperate the user interfaces of our demonstrational multimedia and VE applications.

(3) *Develop Coupling Model:* The coupling model will allow different users to participate in sessions and share semantic and data objects in a secure and consistent fashion. This task will be complete when we can perform all of the couplings required by the demonstration experiments given below.

(4) *Develop Real-Time Model:* The model will perform network and OS scheduling to ensure the real-time coupling requirements of multimedia and VE environments are met. This work will be complete when we can meet the real-time needs of our demonstrational applications.

(5) *Develop Replication/Migration Model:* This model will dynamically replicate/migrate portions of the bus and clients near the sites of users accessing them. This work will be complete when we can establish that the bus is not a central bottleneck and does not have a significant effect on the response times of the users.

(6) *Study and Adapt CORBA Access Control Model:* This model will protect operations on the bus (as opposed to operations on the applications interfaced by the bus). Recently, an access control model has been developed for protecting operations on CORBA objects. We will study this model and adapt it to the collaboration types provided by the collaboration bus. This work will be complete when we have built a mechanism for protecting

operations on all of the new collaboration types defined by the bus.

(7) *Interfacing techniques:* We will identify techniques for interfacing existing software to the bus. In general, we will need to change interfacing software to invoke bus functions. However, when it is not possible to do so, special interfacing agents may be needed to connect the input/output channels of the software to the bus.

(8) *Interconnection Experiments:* We will conduct several experiments (given in the next section) to demonstrate and evaluate the interconnection capabilities of the bus. This work will be complete when we can demonstrate inter-service interoperability among the data, user-interface, coupling, real-time, and replication/migration models and intra-service coupling interoperability for three different coupling systems: XTV, Suite, Shared X.

(9) *Evaluation:* We will evaluate the collaboration bus by determining the range of tools it can interoperate, the amount of effort required to connect a tool to the bus, and performance overhead of using the bus.

(10) *Videos and Reports:* These will document our design, implementation, and evaluation.

B. Project Description

The products and results of this work include general models, specific software for implementing these models, a set of demonstrational interconnections for demonstrating the capabilities of the models, evaluation of the research, and videos/reports describing the above.

As mentioned before, we will develop data, user-interface, coupling, real-time, replication/migration, and bus control models. The data model will include capabilities to define and translate between the data objects of the bus clients (that is, the interoperating applications). In particular, it will include capabilities to support user-defined types and inheritance. It will be portable in that will be supported on multiple heterogeneous operating systems and hardware. As mentioned before, we currently plan to use the Java data model as the basis of our work. The user-interface model will define and translate between the user interfaces of the bus clients. It will include capabilities to define not only traditional user-interface objects such as text buffers, forms, and tables but also non-traditional VE user-interfaces.

The coupling model will link single and multiuser interactive applications so that their users can collaborate with each other. It will include a variety of submodels including session management, awareness, sharing, concurrency control, and access control models. Session management will allow users to enter into joint sessions and will keep track of the various sessions in the system. Awareness will allow users to share information about each other using textual/audio/video streams. Sharing will allow semantic and view objects created for different users to exchange state. Concurrency and access control will protect users from executing inconsistent/unauthorized actions.

All of these models will support a wide range of policies -in particular they will define the policies supported by the target clients we propose to interface. It will support various degrees of sharing including sharing of semantic data, user-interface views, and positions of users within VE views. Moreover, these models will create global policies out of local client-specific policies. For instance, the session management objects will create global session management out of local session management provided by the interfacing clients.

The real-time model and replication/migration models will enable real-time collaboration. The real-time model will perform network and OS scheduling to ensure that real-time coupling needs of multimedia and VE environments are met. This replication/migration model will dynamically replicate/migrate portions of the bus and clients at the sites of users accessing them. For instance, it will replicate real-time services at the sites of the interacting users and migrate a lock object to the site of the user with the lock.

The bus security model will protect the operations on the bus. In particular, it will determine which clients can connect to the bus, which bus services they can use, and which services they can extend.

Infrastructure Software

These models will be implemented in the bus infrastructure. The Java programming language will be used to implement an extendible hierarchy of collaboration objects. For instance, we will define sharing objects to provide various sharing degrees, and multimedia objects to provide video and audio awareness among users. Since the infrastructure will be built on top of Java, it will automatically be portable.

Interconnection Experiments

We will perform a set of interconnection experiments to demonstrate and evaluate the bus. They will demonstrate inter-service interoperability among at least three different services, and intra-service interoperability among at least three different implementations of the same service. Some of the specific experiments we have planned are:

(1) *Interfacing of Heterogeneous Session Management Services:* A global session management facility that keeps track of heterogeneous sessions defined by three different systems: XTV/Suite/Shared X.

(2) *Interfacing of Session Management and Collaborative Applications:* The use of session management of a particular system (XTV/Suite/Shared X) to create sessions between users of ‘‘foreign’’ collaborative applications built using other systems. E.g: the use of XTV session management to create sessions between Suite applications and also between the collaborative VEs described below.

(3) *Interfacing of Heterogeneous Collaborative Applications:* The use of different collaborative applications by different users manipulating the same artifact. E.g: the use of Michigan’s multiuser DistEdit text editor and the Suite multiuser structure editor by different users editing the same document.

(4) *Position Coupled VEs:* Coupling of multiple instances of a VE application, each interacting with a separate user, that allows users to see positions of the other users in their views. Motivation: position coupling of multiple instances of a flight simulation application to simulate a joint sortie and/or combat situation. Each pilot needs to know the positions of other (enemy and friendly) aircrafts.

(5) *Shared View VEs:* Collaborating users see the same view. Motivation: coupling between the displays a pilot and trainer using two different instances of a flight simulator. The trainer needs to see the same view as the pilot in order to tell him what to do.

(6) *Dependent View VEs:* The users have some independence in what they view but their views have dependencies. Motivation: coupling between the displays of crew members of a (single) plane using multiple instances of the flight simulation program. The gunner’s view depends on the actions of the pilot, but he has some flexibility in which parts of the viewable space he actually looks at.

(7) *Fine-Grained Sharing:* Users share changes to selected parts of the views. Motivation: A trainer sees only the instruments panel of the trainee’s aircraft and not the whole aircraft.

(8) *Model Sharing:* Users share a model but not the views. Motivation: The trainer sees a textual view of the instrument panel while the trainee sees a graphical view reflecting the actual display on the simulated aircraft.

(9) *Transaction-coupled VEs:* Delayed transmission of a users actions to other instances of the VE application - the actions of the users are delayed until they commit them in a transaction. E.g: delayed transmission of the actions of users of a CAD application allowing them to collaboratively assemble a tank. Users see the results of others' actions when the latter are "done". Access and concurrency control ensure that users do not perform unauthorized or interfering actions.

(10) *VE-based session management:* The use of a VE for starting collaborative sessions. The space of the VE is populated with icons representing applications to be included in the collaborative sessions. Users can approach an application to start a collaborative session with it. E.g: populating a VE allowing users to move from room to room of an office building. As they enter a conference room, they can start multimedia connections with others in the room, and also interact with a common set of applications.

(11) *Substituting users with agents:* The use of an agent application to simulate the activities of a user. E.g: the use of an application to simulate the actions of a combatant in a flight simulator VE.

In all of these examples, we will be using existing applications to provide the VE, coupling, artifact manipulation, and agent functionality. We have chosen a large number of scenarios involving the use of VE applications since the integration of collaboration and VE is a relatively unexplored, interesting, and difficult area. We have mainly used the flight simulator example to motivate VE. In fact, we will use existing VE applications built at UNC in our experiments. In particular, we will use the state-of-the-art UNC walkthrough application, which is structured similarly to the flight simulation example used above.

C. Technical Rationale

There has recently been significant research in the area of collaboration systems, which has resulted in two kinds of systems: collaborative applications and collaborative tools. The former are special software systems that allow multiple users to share application-specific state. The latter are general purpose systems providing reusable implementations of various collaboration functions. They can be used to share existing, collaboration-transparent, single-user applications or create new collaboration-aware, collaborative applications. A large number of complex issues must be resolved in the design of collaborative tools and applications. As a result, (as shown in section III.E) despite the relative infancy of this area, a great variety of collaborative applications and tools have been developed, which resolve different sets of issues and make different tradeoffs [27, 32, 82].

Thus, we have two ways to support a collaboration infrastructure: i) develop a new collaboration infrastructure, or (ii) develop an integrating infrastructure, which we call the "collaboration bus", that is designed to combine the capabilities of existing collaborative tools and applications. So far, researchers in this area have taken the first approach. In particular, the researchers in our group have developed multiple collaboration infrastructures with different capabilities/implementations - the Suite effort has investigated support for distributed collaborative architectures [23], flexible sharing [21], collaborative undo/redo [13], and protected I/O [91]; the XTV effort [5] has explored how existing X applications can be shared, and the ABC effort [50] has explored how extendible shared window systems can be implemented modularly. These are ongoing research efforts and being carried out as parts of different projects.

We propose to instead investigate the second approach. This approach is attractive because it allows us to support scenarios of the kind given in sections II.D and III.B. It is possible to develop a new system from scratch that support these scenarios by reimplementing the capabilities of the individual systems, but a tool that allows reuse of the implementation of individual capabilities is a much more attractive option because of the complexity of implementing collaborative systems.

As mentioned before, there are are two important forms of interoperability: inter-service interoperability and intra-service interoperability. Inter-service interoperability allows independent services to be composed with each other. Figure 1 illustrates inter-service interoperability. It allows two applications supporting virtual environments user interfaces to be composed with an application providing coupling. The result is a new application supporting collaborative virtual environments. It allows the users of two VEs to couple with each other, where the coupling between the two users is determined by the coupling manager. The previous section gives several examples of scenarios needing inter-service interoperability (e.g. scenarios 2, 5, 10, 11).

Intra-service interoperability allows multiple, competing instances of a service to be used for a single task. Figure 2 illustrates such interoperability. The previous section gives several examples of intra-service interoperability (e.g. scenarios 1 and 3). In this example, the DistEdit and Suite collaborative editors are being used together to manipulate a single document, thereby allowing each set of users to use their favorite editor.

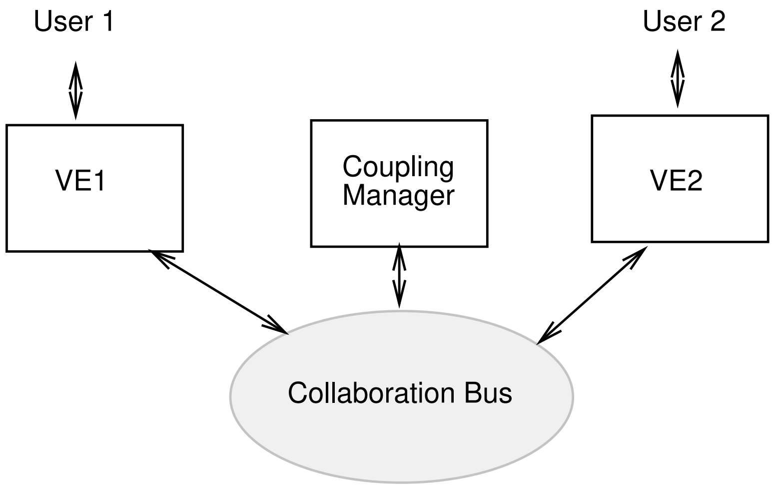

Figure 1: Inter-Service Interoperability

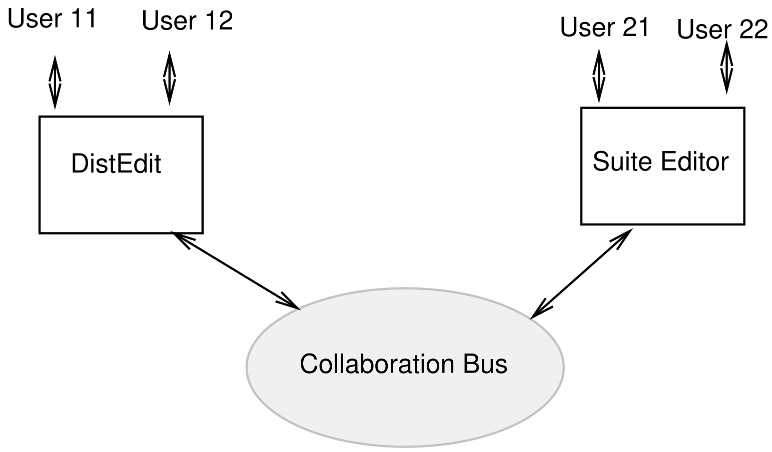

Figure 2: Intra-Service Interoperability

The notion of inter and intra - service interoperability requires an identification of services needed by a collaborative application. These services can be classified into two categories: function services and implementation services. Function services are high-level services that implement the various functions offered by the collaborative system to the user while the implementation services are low-level services used to implement the function services.

Our identification of functions services is based on on the research done as part of the Suite project and by Olson et al [82], which has investigated ways to decompose collaboration into independent functions or dimensions. Like a non-collaborative application, a collaborative application must perform semantic and user-interface functions. In addition, it must couple the state of different users to allow them to exchange information and events. Coupling, in term, can be decomposed into several subfunctions: session management, which allows a group of collaborating users to establish a joint session; sharing, which allows users to link the objects they are working on; concurrency and access control, which prevent users from entering inconsistent and unauthorized commands, respectively; and user awareness, which allows users to be aware of the activities of each other through video/audio/text channels. Figure 3 shows the functional decomposition of a collaborative application.

The main implementation service we shall focus on will be real-time support. This support will be used to ensure that all of the functions mentioned above (Figure 3), working together, can meet the real-time needs of the collaborators.

To explain the role of these services, consider a collaborative CAD application. The semantics of the application define the abstract object being designed, and the user-interface defines the actual user view seen by a user of the application. Session manament determines how the different users establish a joint session with the application. It may require an initiator or chairman to explicitly invite all other participants or it may automatically join all users in the same room in a virtual space into a session. Sharing determines how the semantics and user-interface objects created for different users are linked to each other. It may link only the semantic objects or both the semantic and user-interface objects. Moreover, it may synchronously transmit a user's changes to other users or buffer changes until the sender is ready to transmit them and the listener is willing to receive them. Concurrency control would disallow two users from concurrently manipulating the same part of an object in an inconsistent fashion. It may provide a coarse-grained lock for the whole object or fine-grained locks for different parts of it. Access control would disallow users from manipulating objects they are not authorized to access. It may provide mandatory or control. User awareness would tell each user about the activities of other users. It may establish a video channel between the users allowing each user to observe the gestures of other users or provide special cursors to indicate gestures. Real-time support will allow real-time collaboration among the users of these services. Inter-service interoperability between,

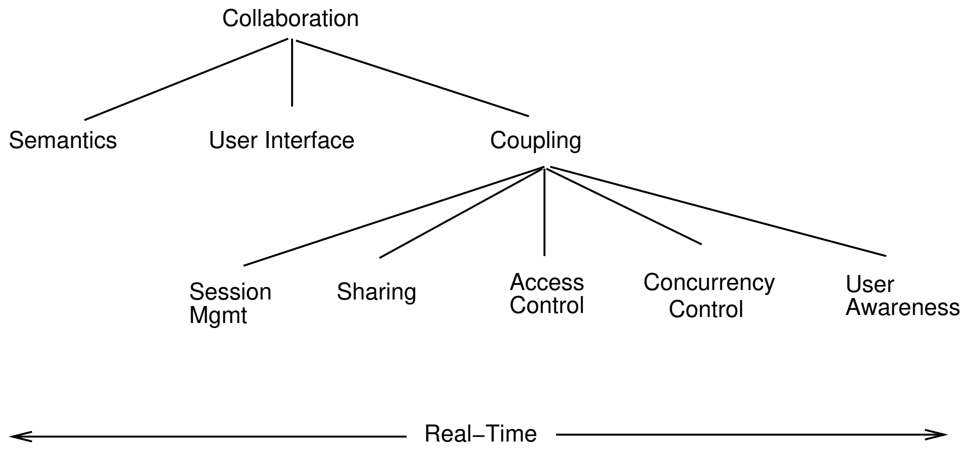

Figure 3: Collaboration Services

say a) real-time support and other services would ensure that all other services - in particular video and VE graphics - meet their real-time deadlines; and b) coupling and other services would ensure that sharing, access control, and concurrency control are provided for all objects - in particular both objects found in both traditional and VE applications. Intra-service interoperability would allow, for instance, systems offering (a) explicit and implicit session management to coexist by mapping events from one policy to another and (b) coarse-grained and fine-grained concurrency control to coexist by choosing a compromise concurrency control (coarse-grained) that can be supported by both systems.

The challenge in providing inter-service interoperability is providing an abstract definition of a service in a bus that can be composed with abstract definitions of other services. So far, the research in collaboration systems has explored specific combinations of services rather in abstract service definitions. For instance, the work on coupling [30] has considered text-based interfaces and developed a parameterized model of coupling that describes all known textual couplings known to us. However, it has not considered graphics, audio, or VE -based interfaces. It is not currently clear how these interfaces must be shared, what events trigger sharing, what it means for concurrent actions on these interfaces to be consistent, what should be the granularity of protection, when and how should users be made aware of the activities of others, what kind of user-interfaces should be provided to specify the protection, consistency, sharing, and awareness parameters, and so on.

Similarly, to date our work on real-time support has has primarily focused on the problem of supporting videoconferencing with digital audio and video. The results of our research have been a number of basic operating system and communication protocol mechanisms for (i) ameliorating the effects of contention for resources in the end- systems and in the network, and (ii) for realizing application-level quality of service requirements. For example, we are presently involved in a collaboration with the Intel Corporation to incorporate the result of this research into a research prototype of the videoconferencing subsystem of Intel's ProShare conferencing product.

However, it is not currently clear what kind of operating and network support should be provided for immersive collaborative virtual environments. We believe that the data generated in a collaborative virtual environment: graphics display lists, object definitions, object position and tracking data, etc. represents a new class of continuous media. Like digital audio and video, these data streams will have delay and loss requirements that will greatly stress operating systems and communication protocols. However, the the magnitude of, and variability in these requirements requires a fundamental rethinking of the problems of management of real-time data in a distributed system. The challenge here is to design small, generic, media adaptations that can sense congestion (possibly with the help of feedback from the receiver) and adapt accordingly. Our current hypothesis is that one can design generic, effective adaptations that scale the media streams (increase or decrease some aspect of the stream) in both the bit-rate and message-rate dimensions.

The challenge in providing intera-service interoperability for a particular service is identifying the design space of the service, and finding a way to translate between systems supporting different parts of this design space. The translation problem is particularly difficult when events in one implementation of the service have no corresponding event in another implementation of it. For instance, a request for a fine-grained lock may have no analogue in a system supporting coarse-grained floor control, an event describing a change to a semantic object may have no analogue in a system that does not distinguish between semantic objects and their views, and an access specification provided by a system supporting discretionary access control may have not be allowed by a system supporting mandatory control.

It is not possible to develop a bus that allows arbitrary applications to interoperate with each other. However, it is our hypothesis that it is possible to develop one that allows a significant fraction if not most of the ‘‘typical’’ collaborative applications to be interfaced with each other. In general, integrating two collaborative systems will require that each be made ‘‘bus aware’’, that is, modified to communicate with the collaboration bus. Once these systems have been modified, however, other systems can be integrated with them without requiring additional changes to these two systems. Thus, the bus, like previous heterogeneous tools such as IDL, would require n translations rather than n square translations if n systems need to be integrated with each other. Often, it is not desirable or possible to change an interfacing client. Therefore, we intend to explore methods for interfacing with the bus that do not require changes to collaborative systems and explore the kind of integration methods that can be supported by these techniques.

While the bus is a central logical structure, it must not be allowed to become a central physical bottleneck. There has been significant work in process replication [17] and migration [60] and we propose to apply it to the concept of the bus. We propose to determine which parts of the bus must be replicated, which must be centralized, which of the centralized objects should be migrated, and what kind of mechanisms and policies should be provided for supporting migration. We will consider migration/replication of not only the parts of the bus but also its clients. For instance, the bus may automatically replicate a VE client on the host of each user wishing to enter a joint session with it. Figure 4 illustrates migration and replication of the bus and its clients. The VE client and the real-time services are replicated while the lock object is a migrating centralized object.

Finally, mechanisms must be developed to secure the bus. These mechanisms will complement the access control provided to protect the operations on the clients interfacing to the bus. Bus

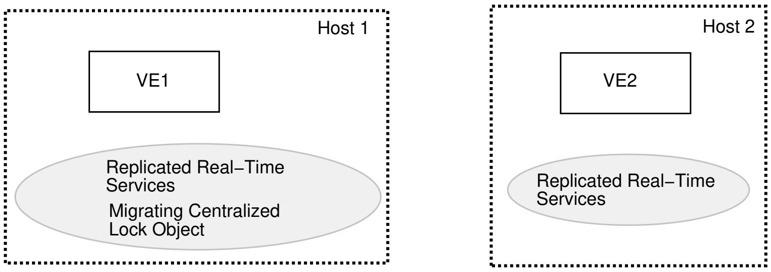

Figure 4: Replication and Migration of the Bus and its Clients

access control will protect the operations on the bus, which may indirectly affect the interfacing clients. For instance, it will protect unauthorized users from adding new coupling managers for connecting two VE interfaces (Figure 1).

D. Technical Approach

We propose to take an iterative approach to testing this hypothesis, wherein we intend to a) identify an initial class of existing systems that are to be integrated, b) identify specific scenarios of integrated use of these systems, c) develop techniques for inter-service and intra-service interoperability; d) implement these techniques in a bus supporting replication, migration, and security; e) provide mechanisms for interfacing with the bus; and f) use the bus to implement the identified scenarios, and g) compare the flexibility, automation, and performance of the bus with existing tools based on the scenarios methods identified in (b). The approach is iterative because our experience in using the software bus and existing heterogeneous tools to build new applications will help us identify new integration scenarios, which in turn will help us redesign the bus.

In the rest of this section, we briefly explain how we propose to take these steps:

Identification of Bus Clients

To make this task tractable, it is important to choose tools with which the researchers are familiar. On the other hand, to make our contributions general, it is important to choose a diverse set of state-of-the-art tools. We have tried to attain a useful balance among these goals by assembling a diverse multi-institutional team that has worked with a variety of tools/services.

The set of tools we have chosen for demonstration purposes are: VE applications developed at UNC, which is doing state-of-the-art research in this area, the Suite collaboration system developed by Prof. Dewan, which supports a wide range of coupling policies for textual interfaces [14, 23, 30, 91]; the XTV system [1], developed at UNC and ODU by Prof. Abdel-Wahab and Jeffay, a widely used shared window system built on top of existing X servers; HP's Shared X system, which is a commercial shared window system built on top of a modified X server [37]; the DIRT system for supporting real-time services for audio and video [46, 99]; and Michigan's DistEdit [66], an Emacs-based collaborative text editor.

This is our initial choice of bus clients, and we propose to finalize this set by the time the project is started.

Identification of Integration Scenarios

In section III.B we outlined the various types of integration scenarios we plan to support: between different session management schemes, between sharing services and VE services, and so on. Once the exact set of tools and the services they offer has been finalized, we propose to identify an initial set of concrete integration scenarios involving the use of these tools. We propose to finish this task in the first three months of the project. It will involve installing the various tools at the sites participating in this project and comparing and contrasting the services they offer. We expect this set to continuously involve as we better understand the integration problem.

Inter Service Interoperability

Inter-service interoperability requires us to generalize the existing semantics of the services so that they can be composed with each other. As mentioned before, we will start with the Java model for defining objects. To define a general user interface model, we will assume that each applications is composed of semantic objects, which describe the semantic data being manipulated; view objects, which define the rendering of these objects on the screen; and subject objects, which define information about the subjects (e.g. name, job, etc), and for VE user interfaces, the positions and orientations of the subjects within a view, A hierarchy of semantic and view objects will be defined to cover the kind of bus clients we propose to support. For instance, we will define view objects that describe rooms within a building (to accommodate MUD/MOO-like use interfaces [18]). Our coupling model will be defined within this framework, allowing users to specify which of these these objects they wish to share (e.g. only the semantic objects, or semantic objects, view objects, and orientations of subjects; when changes to shared objects should be received or sent (e.g. when they make the change or when they enter or leave a room); which kinds of interactions are inconsistent (e.g. changing an object being viewed by another, being at the speaker's podium simultaneously); which kinds of accesses are not allowed (e.g. entering a private office); the information about other users in which they are interested (e.g. their names, their positions within a virtual space); and how they wish to start join sessions (e.g. when they are in the same room, or near each other in a VE).

Our support for real-time interaction will address both network reservation and operating system protocols. Resource reservation in a network is required to ensure reliable delivery of real-time multimedia streams over long distances. We propose to develop a framework for controlling the transmission of media streams via the independent manipulation of bit- and packet-rates. This framework will describe the capabilities of a media generating application as a set of bit-rate 4 message transmission-rate pairs called operating points. Each operating point for a stream will specify a bit and message-rate that the application is capable of generating. The framework characterizes the perceived network environment as a set of feasible operating points which is a subset of the full set of operating points. The elements of the feasible set are operating points which are sustainable under the current network conditions and result in the delivery of media streams with adequate throughput and latency for the application. The application will use feedback from the network to periodically estimate the set of feasible operating points. Based on these estimates, the current operating points is then selected for media transmission.

Our work on OS support is investigating a "new" process model based on the concept of rate-based execution (RBE) that addresses the requirements of this project. In an RBE system, processes specify their desired rate of progress in terms of the number of events they desire to process in an interval of specified duration. A process in an RBE system is guaranteed to make progress at at least its specified rate whenever work exists for the process. We conjecture that the notion of processing rate is a natural and useful abstraction for collaborative virtual environments and we propose to investigate this hypothesis in this project.

Our work on real-time support will take as input the coupling parameters selected by the users, which will determine the QOS (Quality of Service) provided by it. For instance, the processes and messages of users exchanging information incrementally will get precedence over those of users exchanging information on a daily basis.

Intra-Service Interoperability

To develop our techniques for intra service interoperability, we intend to explore commonality in the implementation of this service in various collaboration systems. This work will be based on the research done by Dewan [29] and by Olson et al [82], which has explored the design space of some of these services we plan to support in the bus. We propose to extend this work by identifying the design space for the remaining services, and defining translations among the different points in these spaces provided by our clients system. We expect many of these design spaces to be characterized by hierarchical domains. For instance, session management would be classified as implicit and explicit session management, implicit session management would be further classified according to which actions trigger session management functions: entering a room, editing an artifact, and so on. Translations between different domains will typically involve abstracting operations to a common parent domain. For instance, converting between implicit and explicit session management will involve converting specific operations performed by these schemes to an abstract set of session management operations. When an operation in one domain has no counterpart in the other other, the system may "coerce" the operation to a different form. For instance, it may transform a fine-grained lock request in one system to a coarse-grained request in another and a series of "insert line" events in a text editing user-interface may correspond to a single "insert structure" event in a structure editing user interface. Similarly, if one system allows its users to exchange changes incrementally and another allows its users to exchange them only at the end of each day, then the bus will transmit events to the latter at the end of each day instead of incrementally. It may not always be possible to coerce an operation, in which case it may be disallowed. For instance, if two users are are view coupled, and one is using a structure-editor while the other is using a text editor, then the first user would not be allowed elide a substructure, since such a command is not understood by a text editor and view coupling requires the two users to see the same view.

In the first year, we propose to finish our work on inter and intra-service interoperability for for session management, in the second year for user-interface and networking support; and in the third year for sharing, concurrency/access control, real-time support, and awareness.

Bus Implementation

We propose to provide both kinds of interoperability in a bus implementation that is secure and supports dynamic replication and migration. We will implement the bus in Java, defining hierarchies of Java classes corresponding to the hierarchical service domains mentioned earlier.

In particular, "frequent" bus accesses such as announcements of state changes and real-time support would be implemented in replicated modules executing on the workstation of users; "occasional" bus accesses such as announcements of lock releases would be implemented in centralized, migrating agents, and "rare" bus accesses such as announcements of new users and conferences would be implemented in centralized stationary agents.

We will investigate several policies for migrating bus components and the interfacing clients based on the dynamic behavior of the users. For instance, when new users connect to a session, migratable units might move near or to their hosts, and conversely, when they disconnect from the session, migratable units near them might move to other hosts.

Migration/replication of the bus components will use the mechanisms provided by Java for dynamically loading and instantiating objects. Migration/replication of the clients would, in general, require migration awareness in the clients. However, we intend to also experiment with automatic migration of clients based on logging input events, compressing these events, and then later uncompressing and refeeding the events to new incarnations of the clients at the new site.

As mentioned earlier, CORBA provides a methodology for securing an interoperating agent. Our research on bus security will extend this work by applying this methodology to the objects defined by the bus.

We plan to implement a service in a centralized bus six months after it has been designed. We plan to finish implementation of the replication/migration/security policies in the final year.

Interfacing Existing Systems

As mentioned before, in general, integrating two multiuser systems will require that each be made ‘‘bus aware’’, that is, modified to communicate with the collaboration bus. Often, it is not desirable or possible to change a multiuser system. Therefore, we intend to explore methods for interfacing with the bus that do not require changes to multiuser systems and explore the kind of integration methods that can be supported by these techniques.

So far, we have identified the following techniques: a) No Integration Code: Some integration methods for certain pairs of multiuser systems may not require the writing of new code. For instance, mixing two session management schemes can be done without changing either. b) Unix Pipes: An approach often used in contemporary operating systems for enhancing teletype-based applications is to create a new application that includes the enhancements and interacts with the original application using a UNIX-like pipe. We propose to investigate this approach for teletype-based collaborative systems. c) Pseudo Agents: In our effort on shared window systems, we have had substantial experience with the notion of a ‘‘pseudo server’’, that is, a process placed between a client and a real server that pretends to be the real server and processes requests from the client before forwarding them to the real server. We intend to explore the use of this technique for interfacing between server-based multiuser systems and the bus.

Evaluation

We intend to evaluate the flexibility, automation, and performance of the bus. To evaluate flexibility, we will determine the number of integration scenarios identified by us that can actually be realized in the bus. To evaluate automation, we will determine the amount of bus aware code that needs to be written to interface to the bus. We will also compare this code with the amount of code we have written in our direct interoperation experiments. Furthermore, we will compare the efforts required to interface the client systems using our special collaboration bus and the generic standard Polylith bus used in our VE interconnection experiments. To evaluate performance, we will determine the overhead added by the bus to the response times of the users. An important product of this evaluation will be a set of benchmark collaboration scenarios used in our performance studies.

Once this project this over, a useful follow up project would be to investigate how well it can interoperate other collaboration projects funded under this initiative.

E. Comparison with Other Ongoing Research

There has recently been significant research in the area of collaboration systems, which has resulted in two kinds of systems: collaborative applications and collaborative tools. The former are special software systems that allow multiple users to share application-specific state. The latter are general purpose systems providing reusable implementations of various collaboration functions.

The collaborative applications include CES (Collaborative Editing System) [41], which allows distributed users to edit a document; the RTCAL appointment system [90], which allows users to collaboratively schedule a meeting time; the Colab applications [93], which include an electronic chalkboard and a brainstorming tool; IBIS [102], which supports group decision making by explicitly supporting the notions of issues, positions, and arguments; Information Lens [74], which allows users to filter mail messages; the Coordinator [36], which is based on the speech-act theory of collaboration [36]; the GROVE group editor [32], which allows collaborative, protected editing of a structured outline; the Mercury program development environment [61], which automatically notifies a user editing a module about inconsistent changes made by other users to related modules; Scrutiny [39] and ICICLE [9], which allow a team of software engineers to synchronously inspect a document; FLECSE (FLexible Environment for Collaborative Software Engineering) [26], which includes a collaborative debugger, testing tool, inspection tool, and program editor; the PREP coauthoring system [77, 78], which allows users to add text and audio annotations in multiple columns and offers a flexible algorithm for detecting the differences among different versions of a document; GroupSketch [40] and Tivoli [84], which allow users to sketch unstructured drawings using conventional workstations and Liveboards [33], respectively; GroupDraw [40], Ensemble [81] and GroupDesign [64], which allow users to collaboratively compose structured drawings; the Chronicle spreadsheet [42], which allows users to share named ranges; the MUD [73] and DIVE systems [35], which provide textual and graphical facilities, respectively, to support casual interaction among users; and Video Walls [6] and Shared Awareness Space [31], which provide video connections between different rooms/offices.

The collaboartive tools include the XTV [5], Rapport [34], MMConf [16], Dialogo [72], and ABC [51] shared window systems, which allow a window to be shared by multiple users; the LIZA [38], GroupKit [89], and WeMet [87] toolkits, which provide a library of sharing classes; the DCS system [80], which provides support for some generic conferencing features such as discussion windows and access control; the DistEdit system [65], which allows a group editor to be composed of multiple, existing single-user editors; the MATE system [80], which is a multiuser terminal emulator, the Conversation Builder [62], which supports the speech-act theory of collaboration [36], the Rendezvous [83], GINA [8], and Suite [23] user interface management systems (UIMS), which allow sharing of user-interface abstractions higher-level than windows; and audio servers such as SuiteSound [88] and the system by Angebarnndt et al [7], which provide audio support for multiuser applications.

The main difference betweem our proposed work and these systems is that we propose to address the creation of new collaborative systems by integrating existing systems rather than develop them from scratch. We will use the features in these systems to guide our design of the bus.

The general problem of integrating software systems is not new - this is the classical problem of heterogeneous software systems. Specific instances of this problem have been addressed in programming languages, operating systems, database systems, and software engineering. We are exploring the instance that has arisen from the proliferation of multiuser software systems. By focusing on the characteristics of this specific class of software systems, we expect to identify novel high-level support for integrating members of this class. Nonetheless, we expect to find that existing tools for integrating heterogeneous software are also useful for our target class of systems, primarily because these systems include the properties of other kinds of heterogeneous systems.

Four main kinds of heterogeneity tools have been developed to integrate different kinds of software systems:

a) Heterogeneous hardware tools such as IDL [70], Matchmaker [59], Sun XDR, and the Polylith software bus [85] are designed to allow distributed programs, executing on different hardware platforms to communicate with each other.

b) Heterogeneous language tools such as Polygen [12] support communication between programs written in different programming languages, offering possibly programming paradigms such as the procedural and functional paradigms.

c) Message servers such as Field [86] and YEAST [63, 68] are designed to allow programs to communicate with each other without being aware of each other's identities. Field, in addition, defines a set of events cooperating software development applications would need to communicate among each other.

d) Heterogeneous database tools such as InterBase [10] allow an application to access data from multiple databases. They define universal data models including the data models of existing relational systems and address the problem of composing local transactions on these data into global transactions.

We intend to explore how easily these tools can address these heterogeneity problems arising in multiuser systems. In particular, we intend to explore the use of existing (a) heterogeneous hardware tools to integrate multiuser systems that execute on different hardwares, (b) heterogeneous language tools to integrate systems that are based on different programming paradigms, and c) message servers to allow users to easily switch multiuser systems. We do not propose to directly explore the use of heterogeneous database tools, since none of the current collaborative persistent systems we know store their persistent data in databases. However, we do expect to investigate if we can borrow techniques from these systems to address some of the analogous heterogeneity problems arising in multiuser systems. In particular, these systems have explored the problem of composing global concurrency control from autonomous heterogenous concurrency control mechanisms provided by interoperating data systems. We plan to etend this work by studying how global session management, sharing, and access control can be built from autonomous implementations of these functions provided by the interoperating functions.

F. Previous Accomplishments

Infrastructure and Tools for Collaborative Software Engineering (Dewan)

This project has been funded by an IBM student fellowship, NSF grants IRI-9015442, IRI-9208546, IRI-9408809, and grants from the Software Engineering Research Center, an NSF-sponsored (Grant No. ECD-8913133), Industry/University Cooperative Research Center, funded by IBM, Harris, Bellcore and other industrial affiliates. In this project, we have focussed our attention on high-level support for multiuser applications using collaborative software engineering as a driving problem.

Some of the specific contributions of our previous work on this project include a general, flexible model for coupling the user interfaces of a multiuser application [21, 24, 30]; the abstraction of a shared active variable [23], which integrates and extends properties of program variables, database entities, user-interface views, and versionable objects; a mechanism for associating inheritable attributes with active variables [22]; an application-sharing architecture that combines benefits of the earlier centralized and replicated architectures; a mechanism for creating distributed, persistent objects in conventional operating systems [19, 20]; the abstraction of an audio flow, which allows movement of continuous audio streams across multiple discrete objects [88]; tools for supporting collaboration in all phases of the software engineering lifecycle [28, 75]; implementation of our ideas in a system called Suite; evaluation of Suite; and general principles learned from our experience [25]. In our current project, we are addressing concurrency control, access control [91], merging [76] and asynchronous inspection.

The main product from this grant is the Suite software system, which has been used at UNC, Purdue, University of Minnesota, York University, and University of Kentucky to build and experiment with a variety of collaborative applications.

Real-Time and Multimedia Systems (Jeffay)

This project has been funded by the NSF grant CCR-9110938, and by IBM and Intel. This project considered the class of real-time systems characterized by the existence of strict timing constraints that must be satisfied if the system is to function correctly. We developed a number of new results in the theory of deterministic processor and resource allocation that incorporated more realistic assumptions about program execution constraints. These included analyses of: the problems of scheduling periodic and sporadic tasks non- preemptively [49] the problem of scheduling tasks that communicate and synchronize via shared memory [47, 49] and the problem of accounting for the overhead of interrupt handlers on the guaranteed response times of real-time tasks [56]. The algorithms developed were prototyped in an operating system kernel called YARTOS (Yet Another Real-Time Operating System) [55]. The kernel supports a real-time, predictable-execution, programming model called the real- time producer/consumer (RTP/C) paradigm [57].

To evaluate the formal scheduling models, algorithms, and their realization in the YARTOS kernel, we built a desktop video conferencing system using the kernel [44, 52, 52, 53, 55]. Additional, related research involved the development of real-time operating system support for multimedia applications such as videoconferencing. Results here included a set of adaptations for managing live audio and video streams in a distributed system [45, 48, 58] specifically, delay-jitter management techniques [94, 95] and media scaling techniques for adaptive, rate-based, flow control [100, 101].

Much of the technology developed under this grant has been adopted by the Intel Corporation for the development of their LAN-based videoconferencing product.

Artifact Based Collaboration System (Smith and Jeffay)

This work has been sponsored by NSF grant IRI- 9015443. This project had two primary goals: (a) to build a hypermedia and window system supporting collaboration, (b) to better understand the collaboration process.

To meet the first goal, we developed the Artifact Based Collaboration (ABC) system, [51] whose major system components include: a distributed, hyper-media file system, called the Distributed Graph Storage System (DGS), virtual screen which supports an ABC workspace including conference rooms (windows), ABC browsers and applications, and the Matrix, a layer of operating system infrastructure that provides generic collaborative functions, such as conferencing and hyperlinking, to all browsers and applications that execute within a virtual screen.

We carried out several studies to meet our second goal. Four anthropologists conducted a study of three software development teams over a period of four months [43]. A second study was carried out by a graduate student who worked as a participant observer in one of our project's programming teams over a period of twelve weeks [69]. A third comprised in-depth interviews with members of an established research group, one that had been working together for a number of years [71]. A fourth study analyzed the ways in which a group constructs a body of shared knowledge to underly and support its work. Our studies have contributed to a general theory of computer- mediated collaboration under the concept of Collective Intelligence [92].

The XTV Shared Window System (Wahab and Jeffay)

This project has been funded by ONR and NSF. In this project, we developed the notion of a shared window system, a system allowing logical windows created by a collaboration-unaware client to be shared by multiple users. We implemented this concept in a widely-used system called XTV [4, 15], which in built on top of the X window system. At last count, XTV systems have been imported by over 300 remote users. In particular, our technology has been adopted by the NEC Corporation of Japan for use in their Xprotodist (X Protocol Distributor) product. In addition to the specific system, our work has also identified general issues, problems, and approaches to building shared window systems on top of X [2].

Flexible Shared Windows (Dewan, Jeffay, and Menges)

Unlike Suite and other higher-level infrastructures, an XTV-like shared window system does not give users any flexibility in determining the coupling among their user interfaces. It provides a fixed WYSIWIS (What You See Is What I See) coupling, which ensures that all users see identical views of a window. In this project, we are identifying the semantics, architectures, and implementations of shared windows that semi-automatically and efficiently support the great variety of view sharing methods found in higher-level systems [54]. This project is supported by HP and NSF grant IRI-9508514.

Trellis (Stotts and Nyland)

The Trellis project was funded by NSF grants IRI-9007756 and IRI-9015439. It developed an integrated Petri-net model for specifying hypertext and process control in a collaborative environment. The results include the development of a unique system for multi-reader hypertext and human-interaction called Trellis, and its application to the expression, control, and analysis of coordination protocols for group activities [98]. The system is being used with astronomical images from NASA Goddard to create browsing indexes in which usage protocols (what visualization metjods to employ, to which collaborators to distribute the data, from which collaborators to expect responses, etc) for the images have been encoded.

CAEDI (Stotts and Nyland)

Security (McHugh)

John McHugh has been involved in trusted systems development for over 15 years. He has been involved in the design of a number of secure systems including the Verdix Secure LAN (for which he is currently revising the security policy), TX (a B3 certifiable X window system), the Ford Aerospace trusted file server, and others. He has developed a number of formally based code analysis tools including the Gypsy optimizer and the Gypsy covert channel analysis tool. He recently completed an ARPA funded SBIR contract on compilation checking for assuring correct code and is currently involved in the development of secure Gigabit networks with MCNC under ARPA funding.

Direct Interoperation Experiments (Dewan and Stotts)

Our group has recently conducted several experiments to explore direct interoperation between specific collaboration systems. In particular, we have integrated Suite and XTV [3], Suite and Trellis [97], Suite and the DCS conferencing system built at University of Florida [79], Suite and the DistEdit system built at Michigan [66], Trellis and HTML-browsers, Suite and HTML-browsers, and ABC and HTML-browsers. We have recently started projects integrating Java and Suite, and Java and ABC.

We will expand this research by providing indirect connection, via the bus, among interoperating systems, thereby requiring a single connection from an existing system to the bus rather than a a cross-translation module for every different pair of systems that might want to interoperate.

Indirect VR Interoperation Experiments (Stotts)

This project is being funded by ONR and is exploring how collaborative virtual environments can be composed from existing single-user virtual environments [96]. It is currently using the Polylith software bus [11] for interconnecting the virtual environments. Polylith in an example of a generic technology for interconnecting arbitrary distributed applications. Our work will extend such technology with higher-level abstractions that make it easier to interconnect those distributed applications that support collaboration.

G. Facilities

At the University of North Carolina, we have access to a $15 million system including HP, IBM, Sun, DEC, Silicon Graphics, Intel, and Apple computers, along with some one of a kind large-scale parallel processing machines such as the Pixel Planes 5 graphics supercomputer, and an 8,000 processor MasPar. The machines are interconnected via departmental internetwork composed of FDDI, token-ring and ethernet segments and have access to the Internet. These computers are supported by over 150 gigabytes of magnetic disk space and 40 gigabytes of optical storage. Our primary software environment consists of various flavors of the Unix operating system, the Andrew File System (AFS), and the Motif Windowing System. Languages most commonly used include C, C++, and Prolog.

At HP??

At ODU???

At PSU???

H. Summary of Qualifications

Hussein Abdel-Wahab is a Professor of Computer Science at Old Dominion University. He received a Ph.D. from the University of Waterloo in 1976. He has published several papers in the areas of multimedia collaborative and teleconferencing systems, distance learning and teleteaching, and distributed systems. His research has been funded by NSF, MITRE, IBM, and ONR.

Prasun Dewan is an Associate Professor of Computer Science at the University of North Carolina-Chapel Hill. He received a Ph.D from the University of Wisconsin-Madison in 1986. He has worked in the areas of distributed operating systems, user-interface generation, object-oriented database systems and collaboration systems. His work has been funded by grants from IBM, NSF, and the Software Engineering Research Center, an Industry-University Cooperative Research Center. He has published more than forty papers in refereed journals and conferences and served on the program committes of more than twenty conferences. He has given tutorials on collaborative systems for the NTU satellite program, the IEEE Data Engineering Conference, and the ACM and European conferences on Computer Supported Cooperative Work. He is an Associate Editor of ACM Transactions on Information Systems, Collaborative Computing, Journal of Networks and Computer Applications, and is a Foundation Editor responsible for Group and Organization Interfaces of the Electronic Journal for Universal Computer Science. He is also a member of the IFIP working group WG 2.7 on User-Interface Engineering.

Kevin Jeffay is an Associate Professor of Computer Science at the University of North Carolina at Chapel Hill. He received a Ph.D. in Computer Science from the University of Washington in 1989. He has published over forty papers in the areas of real-time systems, operating systems, multimedia networking, and collaborative systems, served on the program committees of over twenty conferences in these areas, and given tutorials on real-time and multimedia systems for for several conferences. Presently he is program chair for the IEEE Symposium on Real-Time Technology and Applications (RTAS '96) and an IEEE workshop on resource allocation problems in multimedia systems.

John McHugh is Associate Professor and chairman of the Department of Computer Science at Portland State University. He received his Ph.D. in Computer Science from the University of Texas in 1983. He has worked and published in the area of security and software engineering, served on several program committees, and given several tutorials on security. His work has been sponsored by ARPA, NSA, EPA, TRW, and U.S. Army. He won the best paper award at the 1986 International Conference on Computer Design.

John Smith is a Professor of Computer Science at the University of North Carolina at Chapel Hill. He received his Ph.D. from the University of North Carolina in 1970. He has published several papers in writing, hypertext, human-computer interaction, and collaboration infrastructures and theory. His work has been funded by NSF, Army Research Institute, and IBM. He has the Co-chair of the 1994 ACM Conference on Computer Supported Cooperative Work and the Hypertext '87 conference.

Dave Stotts is an Associate Professor of Computer Science at the University of North Carolina at Chapel Hill. He received his Ph.D. from the University of Virginia in 1985. He has published more than fifty papers in the areas of software engineering, visual programming languages, hypertext, and collaborations systems. His research has been funded by ARPA, ONR, and NSF. He has served on several review panels and program committees, is an Associate Editor, Journal of Visual Languages and Computing, is the general chair of ACM Hypertext '96 conference, and was the program co-chair of ACM Hypertext '93 conference.

I. Cost Breakdown

The budget is given in the following pages. We justify below the various components of it.

The number of GRA and PI months for each task in each year are given below.

Milestone	Year1		Year2		Year3	
	Faculty	GRA	Faculty	GRA	Faculty	GRA
Identification of Initial Bus Clients (Smith) and Integration Scenarios	1	12				
Session management and awareness (Wahab)	1	12	1	12	1	12
Data and UI model (Stotts)	1	12	1	12	1	12
Real-Time Services (Jeffay)	1	12	1	12	1	12
Sharing mechanisms (Dewan)	1	12	1	12		
Concurrency and access control services (Smith)			1	12	1	12
Bus replication/migration mechanisms (Dewan)			1	12		
Bus security (McHugh)	1	12	1	12	1	12
Total						

Each milestone corresponds to a function we propose to add in the bus. Included in the work associated with the milestone is the cost of implementing the function, interfacing clients/providers of that service to the bus, performing experiments involving the function, and producing reports and videos involving this work. Next to each milestone we have given the nameof the faculty member who will lead the research of that function.

The cost of one GRA is currently $18,000 annually. We hire students half-time for the 9 months of the academic year, and full-time for the 3 summer months. We have adjusted salaries upward 5% per year, accounting for cost of living and merit raises.

In the first two years of the project, we propose to buy four workstations and disk space. Maintenance fees are computed on hardware as 2.5% of purchase cost in the year acquired, and then 5% of purchase cost in each subsequent year. The Computer Science Dept. charges $310 per person per month for use of the computing facilities the department provides. Fee is escalated 5% in out years.

The travel funds are being requested to allow the researchers to attend ARPA meetings, conferences in this area, and to meet with each other. Research relevant to this project is published in a variety of conferences including the CSCW, ECSCW, CHI, UIST, and Data Engineering conferences. The funds requested will be augmented with University funds of $1000 per year.

Section 1V: Bibliography

[1] Hussein Abdel-Wahab and Kevin Jeffay, "Issues, Problems and Solutions in Sharing X Clients on Multiple Displays," *Internetworking: Research and Experience* 5 (94), pp. 1-15.

[2] Hussein Abdel-Wahab and Kevin Jeffay, "Issues, Problems and Solutions in Sharing X Clients on Multiple Displays," *Internetworking: Research and Experience* 5 (94), pp. 1-15.

[3] H. Abdel-Wahab, S. Makkapati, and P. Dewan, "Merging XTV with Suite for Wider Collaboration," *Second Annual Joint Conference on Information Sciences, Writesville Beach, NC.*, Sept. 1995., pp. 103-106.

[4] H.M. Abdel-Wahab, S.U. Guan, and J. Nievergelt, "Shared Workspaces for Group Collaboration: An Experiment using Internet and UNIX Interprocess Communications," *IEEE Communications Magazine* 26:11 (November 1988), pp. 10-16.

[5] H.M. Abdel-Wahab and M.A Feit, Proceedings, IEEE Conference on Communications Software: Communications for Distributed Applications & Systems, April 1991.

[6] M. Abel, "Experiences in Exploratory Distributed Organization," *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*, Lawrence Erlbaum, 1990, pp. 489-510 .

[7] Susan Angebarnndt, Richard Hyde, Daphne Luong, Nagendra Siravara, and Chris Schmandt, "Integrating Audio and Telephony in a Distributed Workstation Environment," *Proceedings of the Summer 1991 USENIX Conference*, 1991, pp. 419-435.

[8] Thomas Berlage and Andreas Genau, "A Framework for Shared Applications with Replicated Architecture," *Proceedings of Conference on User Interface Systems and Technology*, November 1993.

[9] L. Brothers, V. Sembugamoorthy, and M. Muller, "ICICLE: Groupware for Code Inspection," *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, October 1990, pp. 169-181.

[10] Omran A. Bukhres, Jiansan Chen, Weimin Du, Ahmed K. Elmagarmid, and Robert Pezzoli, "InterBase: An Execution Environment for Heterogeneous Systems," *IEEE Computer* 26:8 (Aug 1993).

[11] John R. Callahan and James M. Purtilo, "A Packaging System for Heterogeneous Execution Environments," *IEEE Transactions on Software Engineering* 17:6 (June 1991), pp. 626-635.

[12] John R. Callahan and James M. Purtilo, “A Packaging System for Heterogeneous Execution Environments,” *IEEE Transactions on Software Engineering* 17:6 (June 1991), pp. 626-635.

[13] Rajiv Choudhary and Prasun Dewan, “Multi-User Undo/Redo,” Technical Report SERC-TR-125-P, Software Engineering Research Center, Purdue University, August 1992.

[14] Rajiv Choudhary and Prasun Dewan, “A General Multi-User Undo/Redo Model,” *Proceedings of European Conference on Computer Supported Work*, October 1995.

[15] Gopeel Chung, Kevin Jeffay, and Hussein Abdel-Wahab, “Accomodating Latecomers in Shared Window Systems,” *IEEE Computer* 26:1 (January 1993), pp. 72-73.

[16] Terrence Crowley, Paul Milazzo, Ellie Baker, Harry Forsdick, and Raymond Tomlinson, “MMConf: An Infrastructure for Building Shared Multimedia Applications,” *Proc. of ACM Conference on Computer Supported Cooperative Work*, October 1990, pp. 329-342.

[17] T. Crowley and H. Forsdick, “MMConf: The Diamond Multimedia Conferencing System,” *Proceedings of the IFIP WG8.4 Groupware Technology Workshop*, August 1989.

[18] Pavel Curtis, “Muddings: Social Phenomena in Text-Based Virtual Reality,” CSL-92-4, Xerox Palo Alto Research Center, 1992.

[19] Prasun Dewan and Eric Vasilik, “An Object Model for Conventional Operating Systems,” *Usenix Computing Systems* 3:4 (Dec 1990), pp. 517-549.

[20] Prasun Dewan and Rajiv Choudhary, “Experience with the Suite Distributed Object Model,” *Proceedings of IEEE Workshop on Experimental Distributed Systems*, October 1990, pp. 57-63.

[21] Prasun Dewan and Rajiv Choudhary, “Flexible User Interface Coupling in Collaborative Systems,” *Proceedings of the ACM CHI’91 Conference*, April 1991, pp. 41-49.

[22] Prasun Dewan, “An Inheritance Model for Supporting Flexible Displays of Data Structures,” *Software—Practice and Experience* 21:7 (July 1991), pp. 719-738 .

[23] Prasun Dewan and Rajiv Choudhary, “A High-Level and Flexible Framework for Implementing Multiuser User Interfaces,” *ACM Transactions on Information Systems* 10:4 (October 1992), pp. 345-380.

[24] Prasun Dewan, “Coupling the User Interfaces of a Multi-User Program,” *ACM SIGGRAPH Video Review, Issue 87*, November 1992. Originally part of the Video Program of *ACM 1992 Conference on Computer Supported Cooperative Work*.

[25] Prasun Dewan, “Principles of Designing Multi-User User Interface Development Environments,” *Proceedings of the IFIP TC2/WG 2.7 Working Conference on Engineering for Human-Computer Interaction, Ellivuori, Finland*, North Holland, August 1992, pp. 35-50.

[26] Prasun Dewan and John Riedl, “Toward Computer-Supported Concurrent Software Engineering,” *IEEE Computer* 26:1 (January 1993), pp. 17-27.

[27] Prasun Dewan, “Tools for Implementing Multiuser User Interfaces ,” *Trends in Software: Issue on User Interface Software* 1, Wiley (1993), pp. 149-172.

[28] Prasun Dewan and John Riedl, “Toward Computer-Supported Concurrent Software Engineering,” *IEEE Computer* 26:1 (January 1993), pp. 17-27.

[29] Prasun Dewan, Rajiv Choudhary, and HongHai Shen, “An Editing-based Characterization of the Design Space of Collaborative Applications,” *Journal of Organizational Computing* 4:3 (1994), pp. 219-240.

[30] Prasun Dewan and Rajiv Choudhary, “Coupling the User Interfaces of a Multiuser Program,” *ACM Transactions on Computer Human Interaction* 2:1 (March 1995), pp. 1-39.

[31] Paul Dourish and Sarah Bly, “Portholes: Supporting Awareness in a Distributed Work Group,” *Proceedings of CHI ’92.*, 1992, pp. 541-547.

[32] Clarence A. Ellis, Simon J. Gibbs, and Gail L. Rein, “Groupware: Some Issues and Experiences,” *CACM* 34:1 (January 1991), pp. 38-58.

[33] S. Elrod, R. Bruce, R. Gold, D. Goldberg, F. Halasz, W. Janssen, D. Lee, K. McCall, E. Pederson, K. Pier, J. Tang, and B. Welch, “Liveboard: A Large Interactive Display Supporting Group Meetings, Presentations, and Remote Collaboration,” *Proceedings of CHI’92*, 1992, pp. 599-607.

[34] J.R. Ensor, S.R. Ahuja, D.N. Horn, and S.E. Lucco, “The Rapport Multimedia Conferencing System: A Software Overview,” *Proceedings of the 2nd IEEE Conference on Computer Workstations*, March 1988, pp. 52-58.

[35] Lennart E. Fahlen, Olov Stahl, Charles Grant Brown, and Christer Carlsson, “A Space Based Model for User Interaction in Shared Synthetic Environments,” *Proceedings of ACM INTERCHI’93*, April 1993, pp. 43-48.

[36] F. Flores, M. Graves, B. Hartfield, and T. Winograd, “Computer Systems and the Design of Organizational Interaction,” *ACM Transactions on Office Information Systems* 6:2 (1988), pp. 153-172.

[37] Daniel Garfinkel, Bruce Welti, and Thomas Yip, "HP Shared X: A Tool for Real-Time Collaboration," *Hewlett-Packard Journal*, April 1994, pp. 23-24.

[38] Simon Gibbs, "LIZA: An Extensible Groupware Toolkit," *CHI'89 Proceedings*, April 1989, pp. 29-36.

[39] John Gintell, John E. Arnold, Michael Houde, Jacek Kruszelnick, Roland McKenney, and Gerard Memmi, "Scrutiny: A Collaborative Inspection and Review System," Tech. Report RAD/USARL/93010, Bull Research and Advanced Development, April 1993.

[40] Saul Greenberg, Mark Roseman, Dave Webster, and Ralph Bohnet, "Issues and Experiences Designing and Implementing Two Group Drawing Tools," *Proceedings of 25th Annual Hawaii International Conference on System Sciences*, January 1992, pp. Vol. 4: 139-150.

[41] Irene Greif, Robert Seliger, and William Weihl, "Atomic Data Abstractions in a Distributed Collaborative Editing System," *ACM Conference record of POPL*, January 1986.

[42] Irene Greif, "Design Group-enabled Applications: A Spreadsheet Example," *Readings in Groupware and Computer-Supported Cooperative Work*, Morgan Kaufmann, 1993, pp. 621-631.

[43] D. Holland, J. Reeves, and A. Larme, "The Constitution of Intellectual Work By Programming Teams," Report # TR92-013., UNC-CH Department of Computer Science Technical.

[44] K. Jeffay,, D.L. Stone, and F.D. Smith, "Kernel Support for Live Digital Audio and Video," *Computer Communications* 15 :6 (July/August 1992), pp. 388-395.

[45] K. Jeffay and D.L. Stone, "Adaptive, Best-Effort Delivery of Live Audio and Video Across Packet-Switched Networks," *ACM Multimedia '94 Video Program, San Francisco, CA, October 1994, 6 mins.*, .

[46] K. Jeffay, "Videoconferencing Systems," *Design and Applications of Multimedia Systems, Vol. 2, Tools and Applications, B. Furht, ed.*, Kluwer Academic Publishers, to appear .

[47] K. Jeffay, "Scheduling Sporadic Tasks with Shared Resources in Hard- Real-Time Systems," *Proc. 13th IEEE Real-Time Systems Symp., Phoenix, AZ*, December 1992 , pp. 89-99 .

[48] K. Jeffay, D.L. Stone, T. Talley, and F.D. Smith, "Adaptive, Best-Effort, Delivery of Audio and Video Data Across Packet-Switched Networks," *Network and Operating System Support for Digital Audio and Video, Proceedings, Third Intl. Workshop, La Jolla, CA, November 1992*, Springer-Verlag, 1993 .

[49] K. Jeffay, D.F. Stanat, and C.U. Martel, “On Non-Preemptive Scheduling of Periodic and Sporadic Tasks,” *Proc. Twelfth IEEE Real-Time Systems Symp., San Antonio, TX.*, December 1991, pp. 129-139. .

[50] K. Jeffay, J. K. Lin, J. Menges, F.D. Smith, and J. B. Smith, “Architecure of the Artifact-Based Collaborative System Matrix,” *Proceedings of CSCW’92*, Nov 1992, pp. 195-202.

[51] K. Jeffay, J. K. Lin, J. Menges, F.D. Smith, and J. B. Smith, “Architecure of the Artifact-Based Collaborative System Matrix,” *Proceedings of CSCW’92*, Nov 1992, pp. 195-202.

[52] K. Jeffay, D.L. Stone, and F.D. Smith, *Kernel Support for Live Digital Audio and Video*, Springer-Verlag, Heidelberg, 1992.

[53] K. Jeffay, “On Kernel Support for Real-Time Multimedia Applications,” *Proc. Third IEEE Workshop on Workstation Operating Systems, Key Biscayne, FL*, April 1992, pp. 39-46.

[54] K. Jeffay, J. K. Lin, J. Menges, F.D. Smith, and J. B. Smith, “Architecure of the Artifact-Based Collaborative System Matrix,” *Proceedings of CSCW’92*, Nov 1992, pp. 195-202.

[55] K. Jeffay, D. Stone, and D. Poirier, “YARTOS: Kernel support for efficient, predictable real-time systems,” *Real-Time Programming," W. Halang and K. Ramamritham, eds.,* , Pergamon Press, 1992., pp. 7-12.

[56] K. Jeffay and D.L. Stone, “Accounting for Interrupt Handling Costs in Dynamic Priority Task Systems,” *Proc. 14th IEEE Real-Time Systems Symp., Raleigh-Durham, NC.*, December 1993, pp. 212-221 .

[57] K. Jeffay, “The Real-Time Producer/Consumer Paradigm: A paradigm for the construction of efficient, predictable real-time systems,” *Proc. 1993 ACM/SIGAPP Symposium on Applied Computing, Indianapolis, IN, ACM Press.*, February 1993, pp. 796-804.

[58] K. Jeffay, D.L. Stone, and F.D. Smith, “Transport and Display Mechanisms For Multimedia Conferencing Across Packet-Switched Networks,” *Computer Networks and ISDN Systems* 26:10 (July 1994), pp. 1281-1304. .

[59] Michael B. Jones and Richard F. Rashid, “Mach and MatchMaker: Kernel and Language Support for Object-Oriented Distributed Systems,” *OOPSLA ’86 Proceedings*, September 1986, pp. 67-77.

[60] E. Jul, H. Levy, N. Hutchinson, and A. Black, “Fine-Grained Mobility in the Emerald System,” *ACM TOCS*, . Originally presented at the *Eleventh ACM Symposium on Operating Systems Principles*, Austin, TX, 8-11 November 1987.

[61] Gail E. Kaiser, Simon M. Kaplan, and Josephine Micallef, "Multiuser, Distributed Language-Based Environments," *IEEE Software* 4:6 (November 1987), pp. 58-69.

[62] Simon M. Kaplan, William J. Tolone, Douglas P. Bogia, and Celsina Bignoli, "Flexible, Active Support for Collaborative Work with ConversationBuilder," *Proceedings of CSCW'92*, Nov 1992.

[63] Simon M. Kaplan, William J. Tolone, Douglas P. Bogia, and Celsina Bignoli, "Flexible, Active Support for Collaborative Work with ConversationBuilder," *Proceedings of CSCW'92*, Nov 1992.

[64] Alain Karsenty, Christophe Tronche, and Michel Beaudouin-Lafon, "GroupDesign: Shared Editing in a Heterogeneous Environment," *Usenix Computing Systems* 6:2 (Spring 1993), pp. 167-195.

[65] Michael Knister and Atul Prakash, "Issues in the Design of a Tolkit for Supporting Multiple Group Editors," *Usenix Computing Systems* 6:2 (Spring 1993), pp. 135-166.

[66] Michael J. Knister and Atul Prakash, "DistEdit: A Distributed Toolkit for Supporting Multiple Group Editors," *Proceedings of ACM Conference on Computer Supported Cooperative Work*, October 1990, pp. 343-356.

[67] Srinivas Konuri, "Telepointer Implementation for a Synchronous Computer Conferencing System," Masters Thesis, University of North Carolina, Department of Computer Science, 1994.

[68] Balachander Krishnamurthy and David S. Rosenblum, "An Event-Action Model of Computer-Supported Cooperative Work: Design and Implementaion," *International Workshop on CSCW*, Informatik Informationen-Reporte 4, 1991, pp. 132-145.

[69] E. Kupstas, "Patterns of Interaction in Same-Time, Same-Place Collaborative Programming.," Tech Report TR93-006., UNC-CH Department of Computer Science.

[70] David Alex Lamb, "IDL: Sharing Intermediate Representations," *ACM Transactions on Programming Languages and Systems* 9:3 (July 1987), pp. 297-318.

[71] M. Lansman and J.B. Smith, Using the Writing Environment to Study Writers Strategies, Computer and Composition, April 92 .

[72] J.C. Lauwers and K.A. Lantz, "Collaboration Awareness in Support of Collaboration Transparency: Requirements for the Next Generation of Shared Window Systems," *Proceedings of ACM CHI'90*, April 1990, pp. 303-312.

[73] B. Liskov, D. Curtis, P. Johnson, and R. Scheifler, "Implementation of Argus," *Proceedings of the Eleventh ACM Symposium on Operating Systems Principles*, 8-11 November 1987, pp. 111-122. In *ACM Operating Systems Review 21*:5.

[74] T. Malone, K. Grant, F. Furback, S. Brobst, and M. Cohen, "Intelligent Information-Sharing Systems," *CACM* 30:5 (May1987), pp. 390-402.

[75] Vahid Mashayekhi, Janet Drake, Wei-Tek Tsai, and John Riedl, "Distributed Collaborative Software Inspection," *IEEE Software* 10:5 (September 1993), pp. 66-75.

[76] Jon Munson and Prasun Dewan, "A Flexible Object Merging Framework," *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, October 1994, pp. 231-242.

[77] Christine M. Neuwirth, David S. Kaufer, Ravinder Chandok, and James H. Morris, "Issues in the Design of Computer Support for Co-authoring and Commenting," *Proceedings of ACM Conference on Computer Supported Cooperative Work*, October 1990, pp. 183-195.

[78] Christine M. Neuwirth, Ravinder Chandok, David S. Kaufer, Paul Erion, James H. Morris, and Dale Miller, "Flexible Diff-ing in a Collaborative Writing System," *Proceedings of ACM Conference on Computer Supported Cooperative Work*, October 1992, pp. 147-154.

[79] R. E. Newman-Wolfe, C. L. Ramirez, H. Pelimuhandiram, M. Montes, M. Webb, and D. L. Wilson, "A Brief Overview of the DCS Distributed Conferencing System," *Proceedings of the Summer Usenix Conference*, June 1991, pp. 437-452.

[80] R. E. Newman-Wolfe, C. L. Ramirez, H. Pelimuhandiram, M. Montes, M. Webb, and D. L. Wilson, "A Brief Overview of the DCS Distributed Conferencing System," *Proceedings of the Summer Usenix Conference*, June 1991, pp. 437-452.

[81] R.E. Newman-Wolfe, M. Webb, and M. Montes,, "Implicit Locking in the Ensemble Object-Oriented Concurrent Graphics Editor," *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, November 1992, pp. 265-272.

[82] Gary M. Olson, Lola J. McGuffin, Eiji Kuwana, and Judith S. Olson, "Designing Software for a Group's Needs: A Functional Analysis of Synchronous Groupware," *Trends in Software: Special Issue on User Interface Software*, Wiley, 1993, pp. 129-148.

[83] John F. Patterson, Ralph D. Hill, Steven L. Rohall, and W. Scott Meeks, "Rendezvous: An Architecture for Synchronous Multi-User Applications," *Proceedings of the Conference on Computer-Supported Cooperative Work*, October 1990, pp. 317-328.

[84] Elin Ronby Pedersen, Kim McCall, Thomas P. Moran, and Frank G. Halasz, ''Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings,'' *Proceedings of ACM INTER-CHI'93*, April 1993, pp. 391-398.

[85] James M. Purtilo, ''The Polylith Software Bus,'' Tech. Report 2469, University of Maryland, 1990.

[86] Steven P. Reiss, ''Connecting Tools Using Message Passing in the Field Environment,'' *IEEE Software* 7:4 (July 1990), pp. 57-66.

[87] James Rhyne and Catherine Wolf, ''Tools for Supporting the Collaborative Process,'' *Proceedings of Conference on User Interface Systems and Technology*, 1992, pp. 161-170.

[88] J. Riedl, V. Mashayekhi, J. Schnepf, M. Claypool, and D. Frankowski, ''SuiteSound: A System for Distributed Collaborative Multimedia,'' *IEEE Transactions on Knowledge and Data Engineering* 5:4 (Aug 1993), pp. 600-609.

[89] Mark Roseman and Saul Greenberg, ''GroupKit: A Groupware Toolkit for Building Real-Time Conferencing Applications,'' *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, November 1992.

[90] Sunil Sarin and Irene Greif, ''Computer-Based Real-Time Conferencing Systems,'' *IEEE Computer* 18:10 (October 1985), pp. 33-49.

[91] Honghai Shen and Prasun Dewan, ''Access Control for Collaborative Environments,'' *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, November 1992, pp. 51-58.

[92] John Smith, *Collective Intelligence in Computer-Based Collaboration*, Lawrence Erlbaum & Assocs, to appear.

[93] Mark Stefik, Gregg Foster, Daniel G. Bobrow, Kenneth Kahn, Stan Lanning, and Lucy Suchman, ''Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings,'' *CACM* 30:1 (January 1987), pp. 32-47.

[94] D.L. Stone and K. Jeffay, ''Queue Monitoring: A Delay Jitter Management Policy,'' *Network and Operating System Support for Digital Audio and Video, Proceedings, Fourth Intl. Workshop, Lancaster, UK.*, Springer-Verlag, , 1994 , pp. 149-160.

[95] D.L. Stone and K. Jeffay, ''An Emperical Study of Delay Jitter Management Policies,'' *ACM Multimedia Systems*, to appear.

[96] D. Stotts and J. Purtilo, ‘‘Virtual Environment Architectures: Interoperability through Software Interconnection Technology,’’ *Proc. of the Third Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, IEEE Computer Society Press, April 17-19 1994, pp. 211-224.

[97] D. Stotts, P. Dewan, J. Munson, and J. Navon,, ‘‘A Three-Level Binding for Collaborative Editing Semantics R. Rada, editor,,’’ *Group Document Management,*, Academic Press.,, to appear 96.

[98] P. David Stotts and Richard Furuta, ‘‘Dynamic Adaptation of Hypertext Structure,’’ *Proceedings of the Third ACM Conference on Hypertext*, December 1991.

[99] T.M. Talley and K. Jeffay, ‘‘Two-Dimensional Scaling Techniques For Adaptive, Rate-Based Transmission Control of Live Audio and Video Streams,’’ *IEEE/ACM Transactions on Networking*, to appear .

[100] T.M. Talley and K. Jeffay, ‘‘Two-Dimensional Scaling Techniques For Adaptive, Rate-Based Transmission Control of Live Audio and Video Streams ,’’ *Proc. Second ACM International Conference on Multimedia, San Francisco, CA*, October 1994 , pp. 247-254.

[101] T.M. Talley and K. Jeffay, ‘‘Two-Dimensional Scaling Techniques For Adaptive, Rate-Based Transmission Control of Live Audio and Video Streams,’’ *IEEE/ACM Transactions on Networking*, to appear .

[102] K.C. Burgess Yakemovic and E. Jeffrey Conklin, ‘‘Report on a Development Project Use of an Issue-Based Information System,’’ *Proceedings of CSCW’90*, October 1990, pp. 105-118.