INTERACTION STREAMS: - AN APPROACH FOR WORKSPACE MANAGEMENT IN COLLABORATIVE ENVIRONMENTS

by

PREETI MEHRA

A thesis submitted to the

Graduate School-New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

written under the direction of

Prof. Manish Parashar

and approved by

____

____

____

New Brunswick, New Jersey

October, 2003

ABSTRACT OF DISSERTATION

Interaction Streams: - An Approach for Workspace Management in Collaborative Environments

by Preeti Mehra

Dissertation Director:

Professor Manish Parashar

A Collaboratory has been defined as a group of people working together from diverse physical locations on a common task using a shared workspace. A computational collaboratory is a virtual environment where scientists and researchers work together to solve complex interdisciplinary problems, despite geographic and organizational boundaries. These systems provide uniform pervasive (and collaborative) access to computational resources, services, applications and/or data, and can expand the resources available to researchers, enable multidisciplinary collaborations and problem solving, accelerate the dissemination of knowledge, and increase the efficiency of research. The emergence of Grids-based computational collaboratories and the potential for seamless aggregation, integration and interactions has made it possible for scientists and engineers to conceive a new generation of realistic, scientific and engineering simulations of complex physical phenomena.

In this thesis the collaboration requirements for a computational collaboratory are investigated. These requirements are based on the nature of interactions between users and their interactions with the applications, services and data. The design and implementation of a workspace management and organization tool called **Interaction Streams** for private/shared workspaces in portals of collaborative environments is described. Interaction Streams maintain a navigable record of all user-user and user-applications interactions and collaborations. Each user has a personal Interaction Stream consisting of events occurring in their local and shared workspaces. Stream filters are provided to organize and present the information from these streams. Visual representations of these streams consist of a stack of time ordered documents depicting the events. Features such as borders, titles and icons complement the documents based on the keys to ensure rapid accessibility and categorization. Streams hold answers to questions such as “did a group of users ever collaborate”, “on which applications or in what context did a group of users collaborate”, or “which interactions were done by a particular group of users and on which applications”.

The portals are a part of the Discover computational collaboratory [1]. Discover is a virtual, interactive computational collaboratory that enables geographically distributed scientists and engineers to collaboratively monitor, and control high performance parallel/distributed applications on the Grid. Its primary goal is to bring Grid applications to the scientists'/engineers' desktop, enabling them to collaboratively access, interrogate, interact with and steer these applications using pervasive portals.

Acknowledgements

I am grateful to my advisor Prof. Manish Parashar for his invaluable guidance, immense patience, encouragement and support throughout my stay at Rutgers. I am thankful to Prof. Deborah Silver and Prof. Yanyong Zhang for their valuable advice and pertinent suggestions regarding my thesis. I would also like to thank the CAIP support staff for their prompt and detailed responses to my queries and for the excellent facilities that they provide in the various laboratories at CAIP and in particular at The Applied Software Systems Laboratory (TASSL). I acknowledge the support and love of all my friends for making my studies at Rutgers a memorable phase of my life. Finally, this work wouldn’t have been possible without the love, support and encouragement from my family members.

Table of Contents

Abstract of Dissertation ………………………………………………………………i

Acknowledgements…………………………………………………………………...iv

Table of Contents ….……………………………………………………………………v

List of Tables…….……...…………….……………………………………….……viii

List of Figures. ………….………………………………………….…...…………....ix

1. Introduction. ………….………………………………………………………...1

1.1 Objective ……….…………………………………………………………1

1.2 Background …….…………………………………………………………1

1.3 Problem Statement…..…………………………………………………….2

1.4 Contribution to thesis……………………………………………………...3

1.5 Organization of the thesis…………………………………………………4

2. Background and Related Research…………………………………………………5

2.1 Background work……………………………………………………….……5

2.2 Related Work………………………………………………………...…8

2.2.1 Lifestreams………………………………………………..………...8

2.2.2 Mirror World…………………………………………………...........11

2.2.3 DISCIPLE…………………..……………………………………..12

3. DISCOVER- Computational Collaboratory for Interaction and Steering …….15

3.1 Client in DISCOVER……………………………………………………….16

3.2 Client Server Communication Model………………………………………17

3.3 Multiple Application Support………………………………………………19

4. Interaction Stream in DISCOVER ……………………………….………………………22

4.1 User Interaction Stream ……………………………………………………22

4.1.1 User Interaction Stream Views……………….……...…………………23

4.1.2 User Interaction Stream Views…………………...…….…………25

4.1.3 User Substream Views ……………………………..…………..25

4.2 Application Stream ……………………………………………………...28

4.2.1 Application Stream View……………………………………….…28

5. Design of the Interaction Streams ………………………………………………...30

5.1 Elements Comprising the Interaction Stream…………………………..….30

5.1.1 Classification of events in User Workspace……………………....31

5.2 Client Server Communication ……………………………………….…...32

5.2.1 DISCOVER system Architecture……………………………………33

5.2.2 Communication Sequence of Client Server…………………………35

5.3 Issues Faced……………..37

5.3.1 Stream Logging ……………………………………………………...37

5.3.2 Stream Retrievals……………………………………………………38

6. Implementation of Interaction Streams……………………….…………………..…41

6.1 Implementation………………………………………………………….41

6.1.1 Stream Retrievals ……………………………………………….....41

6.1.2 Stream Logging Implementation……………………………………42

6.1.3 Database Design…..………….……………………………………43

6.1.4 Applications Used for testing……...………………………………45

6.2 Evaluation …………………………………………………………………45

6.2.1 Experiment 1 …………………………………………………………...46

6.2.2 Experiment 2……………………………………………………...….47

6.2.3 Experiment 3…………………………………...……………….47

7. Conclusion and Future Work …………………………………………………..49

References………………………………………………………………………….51

List of Tables

1. Categories of events in the user Workspace ……………………………………32

List of Figures

1. Entities and their Interactions……………………………………………………..6

2. The Lifestreams Viewport for X Windows……………………………………….9

3. The Eventstream interface of DISCIPLE………………………………………..13

4. Three tier architecture of DISCOVER…………………………………………...16

5. Client Server Communication Model in DISCOVER…………………………...18

6. Snapshot of the DISCOVER portal ...…………………………………………...20

7. Snapshot of DISCOVER portal with multiple Application Support…………..…21

8. Collaboration Stream view in DISCOVER……...……………………………....24

9. Substream View in DISCOVER………………………………………………...26

10. The system architecture of DISCOVER……………………………………....34

11. Client Server communication with Collaboration ………………………………36

12. Collaboration Stream View for zoom in feature

    a. Initial view of the requested stream ……………………………………..39

    b. The stream view after the activation of the zoom in feature……………..40

13. Entity relationship diagram of Discover database………………………………44

14. Graph depicting time overhead for insertions ……………………….…………..46

15. Graph depicting memory usage for streams ……………………………………47

16. Graph depicting retrieval times for streams………………………….………..…48

Chapter 1

Introduction

1.1 Objective

The objectives of the thesis are to:

- Identify the set of requirements for making a groupware system usable.
- Develop a feature called the Interaction Streams that will meet the requirements of such a system and improve its usability.
- Design, implement and evaluate the prototype of this feature by its incorporation in to the Discover Computational collaboratory to enable session archival and logging of the collaborators and the applications.

1.2 Background

A collaborative environment is a groupware system where geographically distant users meet to solve the task at hand. The advancements in technologies have aided the computer collaborative work experience come closer to the real world collaborative experience. The technology designed to facilitate the work of groups by helping them communicate co-ordinate and solve problems is coined as groupware. “The field of study that examines the design adoption and use of a groupware is called as the Computer Supported Collaborative Work (CSCW).” A collaborative system can be categorized into two primary types based on the task that they are solving namely Synchronous systems (distant users working together in real-time in the environment) and Asynchronous systems (distant users working on their own convenient time in the environment).

A lot of importance is put into the design of a real time collaborative groupware. The design not only requires the immense knowledge of the group's behavior but also the knowledge of the underlying technology that affects the quality of the system. Due to the dynamic nature of a synchronous collaborative groupware, the key features built into the system should include scalability, responsiveness, consistency and usability. "Usability is the quality of a system that makes it easy to learn, easy to use, easy to remember, error tolerant and subjectively pleasing." [13] The success of a shared environment is primarily determined by the usability of the system. A collaborative system can be usable only if it meets requirements of unconstrained operation, ability to work in functional real world groups and offers response times comparable to single user systems. The ease of use of a system can be achieved by placing the least amount of restrictions on the workspace of the user and by the provision of various means and tools to store, manipulate, visualize, annotate and share information and data.

1.3 Problem Statement

During the design and the development phase of collaboratory, maximum efforts are put to form a robust infrastructure so that it can provide real-time communication for the geographically distant users. Even though this infrastructure is the most indispensable part of the system, the usability of the system is restricted to the tools provided to the user on the front end of the environment. Most of the collaborative environments provide tools for sharing files, messages, voice applications, images etc… These applications however suffer from poor session and application logging and retrievals. Session logging and retrieval tools have proved to be very useful feature in a collaborative environment. Such tools aid in the visualization and review of the project ranging from textual to graphic representations of the events that occurred during the project. A great amount of data can be summarized with ease aiding any member of the group to get up to date with the current status of the project.

While such tools can increase the usability of the collaboratory, developing them poses a lot of challenges. Data storage, data retrieval, data representation, scalability, and privilege checks are a few of them. A key to the development of a collaborative environment is its ease of use. A lot of thought has to be put into the design and development of the front end of the collaboratory so that it provides the user with the maximum intended functionality without imposing restrictions on its use.

1.4 Contribution to the Thesis

This thesis addresses the issues faced by the users of a collaborative environment and makes the following contributions: -

- It lists the issues faced and formulates the requirements of a collaborative system to deal with these issues, hence increase the usability of the system.
- Design and implementation of a tool used for session logging and retrieval that aids in the visualization and review of the collaborative environment.
- Incorporation of this tool into DISCOVER – a web based computational collaboratory. Evaluation of the tool in this environment
- Enabling multiple application access to the clients in DISCOVER by multi-threading the web based clients.

1.5 Organization of the Thesis

This thesis has been organized into seven chapters. The background work and the related work comprise the main body of Chapter 2. It introduces certain projects that influence our approach to increase the usability of the system. It then compares our approach to these approaches.

Chapter 3 describes DISCOVER – A web based Computational Collaboratory for interaction and Steering. This also contains the design overview of the system with special emphasis on the client architecture and the client server communication channel. It then introduces the concept of multi application support for the client and states its design.

Chapter 4 describes the design of the Interaction Streams for the DISCOVER system. It further describes the implementation and the operation of these streams. Chapter 5 and 6 describes the design and the implementation and evaluation of the tool in the DISCOVER system. Chapter 7 presents the conclusions and has directions for its future work.

Chapter 2

Background and Related Research

2.1 Requirements of Collaborative Portals

The aim of a collaborative computational problem solving environment is to provide scientists with a set of mechanisms and tools for collecting, storing, manipulating, accessing and sharing information using meaningful annotation tools supporting their views and ideas. When accessing such an environment, a scientist should be able to reliably and transparently initialize, deploy, access, monitor and steer applications while collaborating with other scientists. Central to the design of such a collaborative scientific problem solving environment is understanding the types of interactions taking place between the participating entities and, identifying their corresponding enabling tools and mechanisms. Global scientific investigation involves seamless interactions between all the participating entities – i.e. users (scientists and engineers), data sources, applications, and resources. User to user(s) interactions can be defined as a sharing of any human-understandable information, being textual, visual or audible, among a managed group of interested parties. Applications interact with resources and data centers when they need to allocate resources and fetch data to perform a certain computation or to store computation results. Users interact with applications, data and resources to monitor, control and steer the application, and resources and explore the results produced. Based on these interactions a set of tools for enabling collaborative computational problem solving are described below.

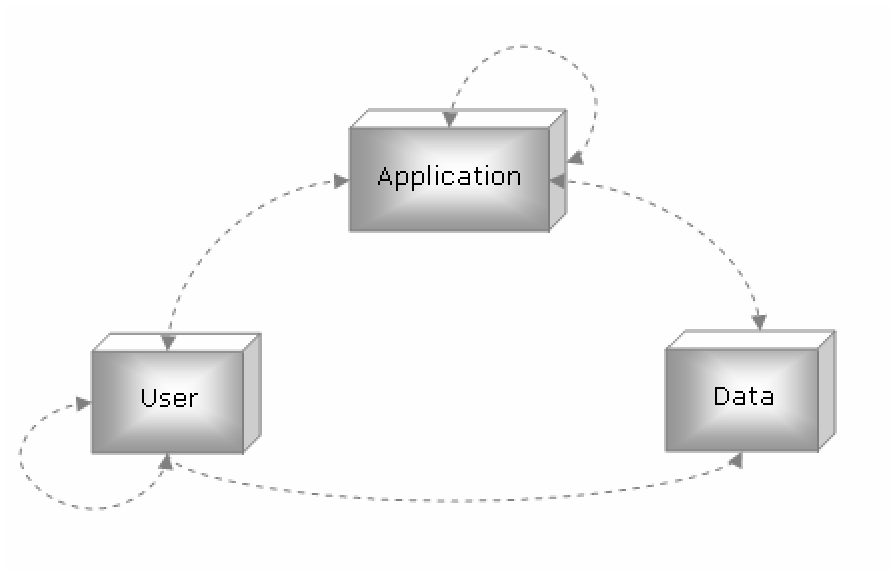


Figure 1: Entities and their interactions in collaborative environment

Group management allows users to create or join groups by interest. Every group needs to be maintained by administrators who have special privileges for altering the group configuration, such as the credentials and capabilities required for joining a group, privileges of each group member and total number of members allowed.

Monitoring and steering allows users to (synchronously or asynchronously) access, monitor and control application objects in a secure, controlled and consistent way. **Sharing tools** associated with the results from queries allow users to share specific results with a group of scientists. **Data manipulation tools** allow users to render the results obtained to enhance their analysis and interpretation.

Locking/leasing mechanisms are necessary to maintain consistency when an application is accessed concurrently by multiple users but can only support one control channel. A lease is a lock that is valid for a fixed duration. Locks/leases are only granted to users that have appropriate privileges.

Collaboration tools such as **Chat and Whiteboard** provide users with a means for collaborating while analyzing the results. The whiteboard also provides **annotation tools** allowing users to mark, draw, write or point to information on rendered application results.

Logging and Management tools are necessary for maintaining history of the sequence of events that occurred in the system, including users' requests, application responses or status messages and user collaborations. These tools are critical for allowing scientists to analyze and interpret results of interactions and collaborations.

A key focus is the design of an effective workspace management tool to provide a navigable record of all user-user and user-applications interactions and collaborations in computational collaboratories. General collaborative environments such as Groove [16] or Netmeeting [17] provide users with tools for sharing files, text/voice/video messages and applications. However, this is done in a generic manner without taking the specific requirements of an application into account and providing a set of standardized tools to the collaborating users. Recent scientific collaboratory efforts include the Upper Atmospheric Research Collaboratory[18], providing an environment for space physicists around the globe to collaborate and share information collected by upper atmospheric instruments. The Astrophysics Simulation Portal[19] built on Cactus[20] is a framework that allows scientists to deploy and monitor simulations on the computational Grid using the Globus[21] toolkit. While these frameworks address collaboration, their tools are highly customized to the application domain. Most of these collaborative environments suffer from poor session logging and retrieval tools. Some environments do support such tools but are limited to logging in shared workspaces and used for generating session replays or enabling asynchronous work in the collaboratory. A few of them do provide information retrieval and querying on the logged data but are almost unusable due to the complex nature of the tools provided. Hence Interaction Streams is developed in order to enable logging, information retrieval and representation of the workspaces in the computation collaboratries.

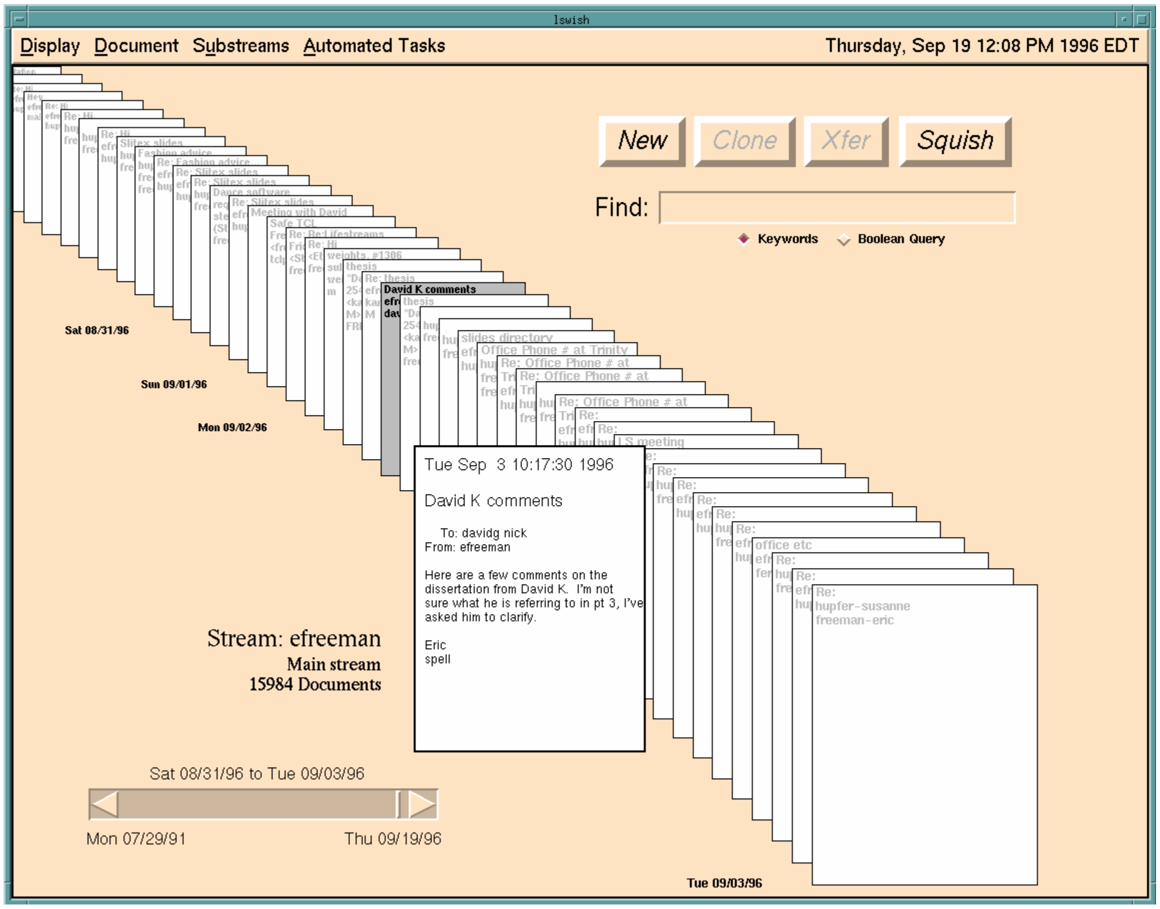
2.2 Related Work

Design of a groupware system has been a field of continuous research. It requires a complete understanding of its prospective users. This section describes a few systems that present personal workspace (desktop) and shared workspace organization metaphors.

2.2.1 Lifestream

Developed at the Yale University by a team led by Dr. Gelernter, Lifestreams, a concept for dynamically organizing a user's personal workspace was introduced. Lifestreams uses a simple organizational metaphor; a time ordered stream of documents to replace the conventional files and directories [3][4][5]. Stream filters and software agents are used to organize, locate, summarize and monitor incoming information. Lifestreams has also been termed as the diary of ones electronic life. The lifestream of a user contains every document that he creates or the documents that are sent to him by other users. The tail of the stream contains documents from the past (starting with ones electronic birth certificate). Moving away from the tail and towards the present, the stream contains more recent documents. The future of the stream contains documents that one might need: reminders, calendar items, and to-do lists.

The lifestreams project is based on a client-server architecture that runs over the internet. Server is responsible for the handling and the storage of the documents. The client of the system is a viewport that enables the basic operations associated with the streams by providing an interface to it. Figure (2) depicts the viewport developed for the X windows system. The figure 2 shows a stream of documents that can be rolled into the past by a date annotated scroll bar. Colors and animation are extensively used as a means to depict the salient features of the document and provides an easy means to distinguish the documents.



Tue Sep 3 10:17:30 1996

David K comments

To: davidg nick
From: efreeman

Here are a few comments on the dissertation from David K. I'm not sure what he is referring to in pt 3, I've asked him to clarify.

Eric
spell

Figure 2: The Lifestreams Viewport for X Windows

The lifestreams system provides 5 basic operations.

1. New: - This operation is used when the user creates any new documents ranging from an email to calendar to do list.
2. Clone: - This operation is used to create the duplicates of the documents
3. Xfer: - This operation is used when one needs to forward documents or emails
4. Find: - This operation allows user to enter a Boolean query and creates a substream.
5. Summarize: - This operation allows the user to take a substream and compress it into an overview document.

The operation that gives lifestreams the most power is the find operation where one can enter a Boolean query. This query results in a substream of documents satisfying the stated criteria. Substreams can be nested thus adding some complexity to the tool. As new documents satisfying the criteria are created, they get added to the stream. No extra storage is provided for this sub stream hence these Boolean queries can be saved and recovered whenever needed. The Boolean queries can range from a simple query like all emails received to complex queries like emails received on the subject A on date B.

Summarize also termed as the squish technology is used to generate summary of substreams. It does so by creating the summary as a document and adding that document to the particular substream. The task of classifying the documents into preset categories is accomplished by the use of agents. There are a variety of agents that are spawned based on the kind of document under consideration. Lifestreams uses three kinds of agents that extend the functionally provided.

The agents are listed below.

1. Personal Agents: - These are typically attached to the user interface and can automate tasks.
2. Document Agents: - These live on the documents and are spawned by various events like the access of the document.
3. Stream Agents: - These agents are attached to the streams and are spawned when stream undergoes some change like a new document is added to the stream.

Lifestreams has proved to a very useful tool for managing and accessing ones personal information. We feel that the concepts embodied by lifestreams can be adapted to aid in the visualization of the collaborative sessions in DISCOVER.

2.2.2 Mirror World

Fueled by the research and writings of Dr. David Gelernter and Dr. Eric Freeman, Mirror Worlds Technologies was established in 1997 to develop and commercialize the patented technology of TiTLE[8]. The first commercial release of their product Scopeware enterprise solution was launched in March 2001. A desktop product, called the Scopeware Vision was later launched in 2002. Scopeware is a visual management system that resides over the operating system and the native file system still using their time based ordering. Scopeware Vision is distinguished from typical search engines in how it presents the information to you based on the query. The stream is still a time-ordered visual representation of all the information, normalized in a single view. Files and email appear together in a browsable, or searchable, view allowing for quick visual recognition.

The Vision stream, as compared to browsing or searching traditional file folders, is a more visual and comprehensive [8].

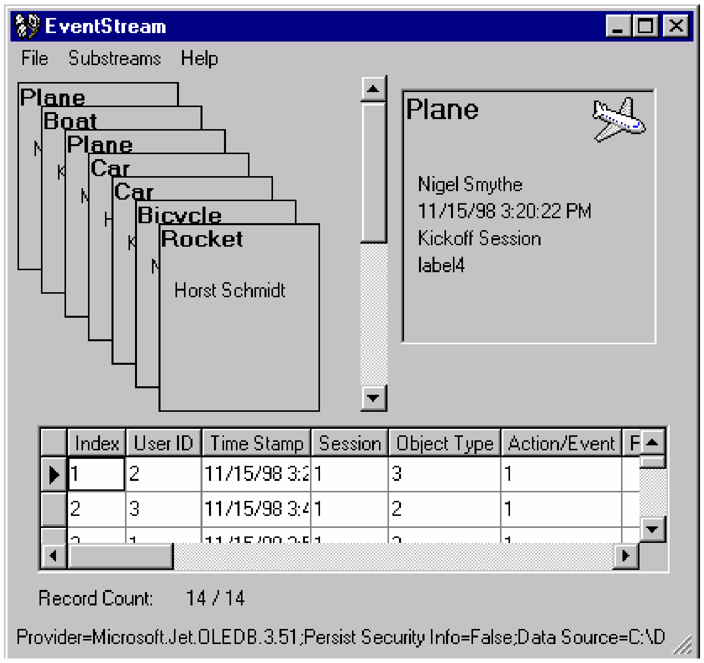
Scopeware Vision's patented TiTLE system represents the four key dimensions that mirror the way a human mind works. The four key dimensions are listed below.

1. Time: - People think in time ordered sequence and vision displays information in a time ordered manner.
2. Type: - Different types of information like the email, spread sheets etc are distinguished by the use of visual representations.
3. Look: - Thumbnail representations are provided for rapid recognition.
4. Essence: - Rather than launching the application to view a document, vision provides a summary of the document.

2.2.3 DISCIPLE

Developed at The Rutgers University, DISCIPLE (DIstributed System for Collaborative Information Processing and LEarning) is a project in mobile computing and collaboration. The key objective of the DISCIPLE project has been to develop an advanced groupware design that enables interactive collaboration in the context of the task at hand. The participants at different locations collaboratively access, manipulate, analyze, and evaluate multimedia data. [6]

The whiteboard in DISCIPLE enables users to work and see the current state of the objects in their current project. Inspired by the Lifestreams project, a similar tool called Eventstreams was added to the project to add the functionality of data retrieval and access. The streams are composed of events that are actions performed on the objects in the shared space. The concept of personal document stream has been expanded to define a collaboratory's Lifestreams of events. All the users in a group are considered to be of equal privilege and the whole work is done in a collaborative mode.



Nigel Smythe
11/15/98 3:20:22 PM
Kickoff Session
label4

	Index	User ID	Time Stamp	Session	Object Type	Action/Event	F
▶	1	2	11/15/98 3:2	1	3	1	
	2	3	11/15/98 3:4	1	2	1	

Record Count: 14 / 14

Provider=Microsoft.Jet.OLEDB.3.51;Persist Security Info=False;Data Source=C:\D

Figure 3:- The Eventstream interface of DISCIPLE

Figure 3 shows a view of the Eventstream interface. [7] It lists the events triggered by the members of the shared workspace on different objects. Eventstream provides the functionality of the Boolean query of the find operation by providing drop down menus and prompting the user to make a selection from this. Eventstream is extensively used as a logging tool to get the collaborators up to date with the current activity. Even though the concept of lifestreams has been well used as eventstreams in a collaborative environment of DISCIPLE, it does fall short in many scenarios for the DISCOVER project. Every user has preset privilege levels for every application. Thus the idea of a single Lifestream for all the users does not prove to be attractive. In our approach, we decided to stay with the idea of an every collaborator having his personal stream, but as the project is a collaborative environment, we enabled the union and intersection of the streams strictly monitored by the collaborators privilege levels for every application.

Chapter 3

DISCOVER – Computational Collaboratory for Interaction and Steering

DISCOVER is an interactive computational collaboratory that provides transparent access to remote complex applications for monitoring and sharing. The main objective of this application is to provide geographically distributed scientists to collaborate monitor and control the application by providing a user with a web-based user-friendly interface. This chapter provides an overview of the design of DISCOVER ranging from the server to the virtual desktop. It then introduces the design of the thin clients to enable multiple application access and control as opposed to single application access.

DISCOVER is designed as a three tier architecture (fig 4), comprised of remote thin detachable client portals at the front end, a peer to peer network of interaction servers in the middle and a control of network sensors, actuators and interaction agents superimposed on the application at the backend. [2][3]

The client can be invoked from a web browser. The client when connected to the server receives a list of active running application. The current client has a capability to monitor one application per client. The middle tier is a web server that extends the capability of interaction and collaboration. This server is responsible for translating and forwarding the requests and steering commands from the clients to the control network with the applications at the backend. Some of the other functionalities built into the server are session management, locking mechanism, concurrency control, and security and authentication services.

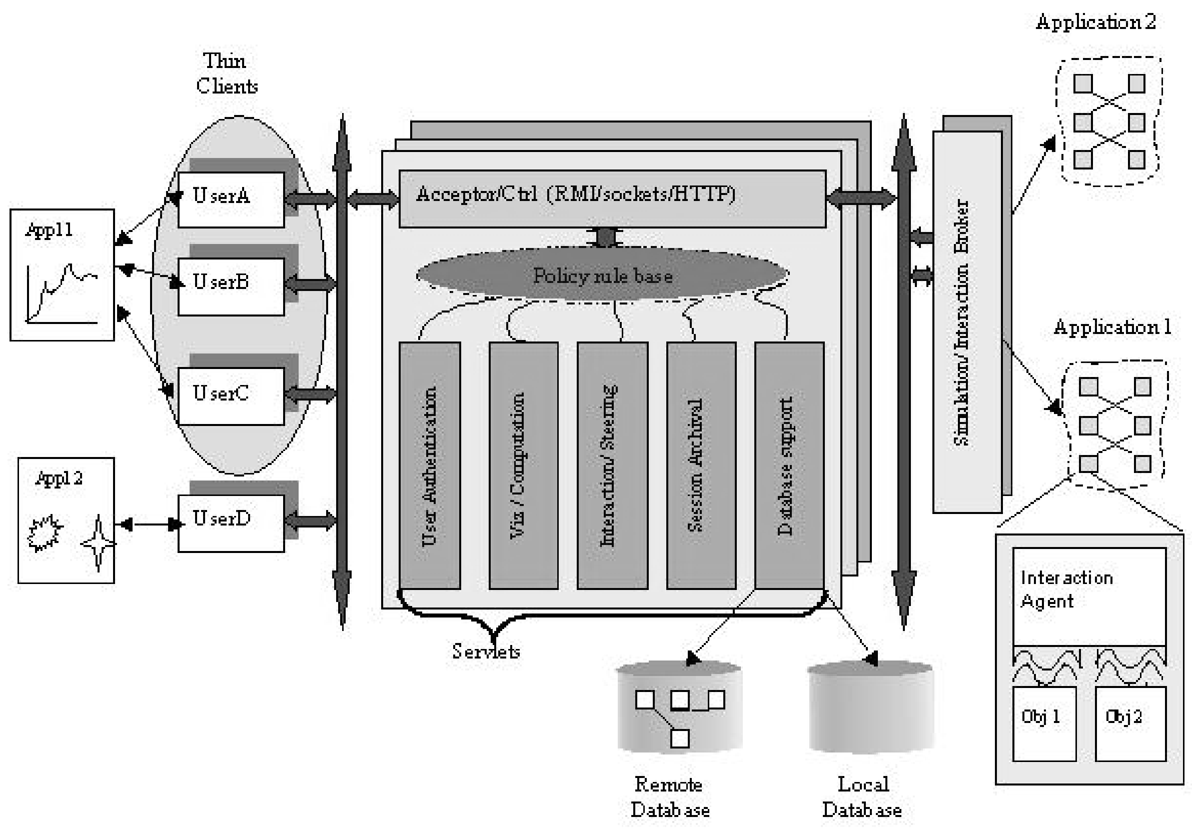


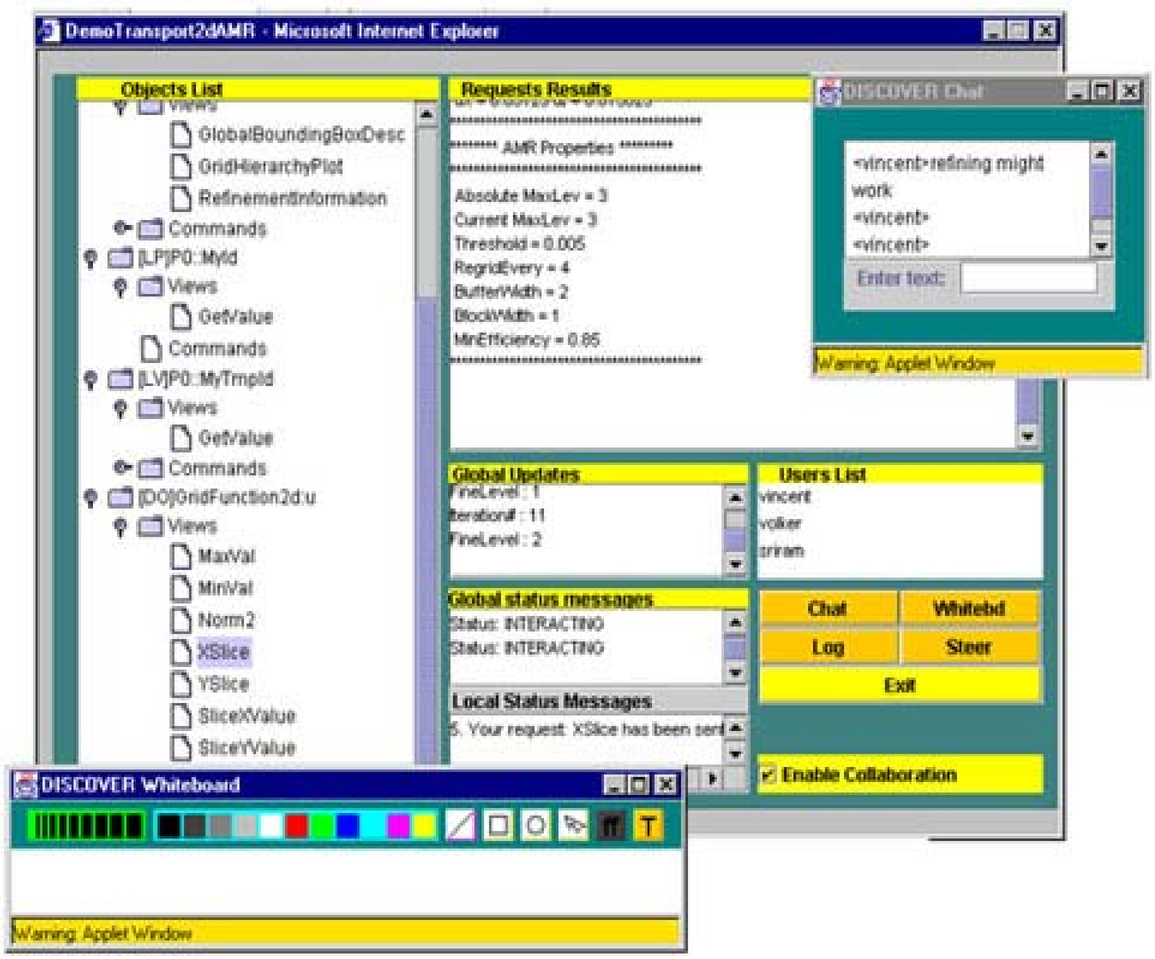
Figure 4: - Three tier architecture of DISCOVER

3.1 The Client

The client portal is a virtual desktop that provides means for the scientist to monitor and interact with the applications, collaborate with his group members and review the logs of the information. The desktop is divided into two domains called the global and the local desktop. A global desktop is a shared workspace of the group formed by the collaborators. DISCOVER employs replicated shared workspace architecture for efficient response times. Locking mechanisms are incorporated to deal with the issues of replicated workspaces. Objects can be dragged into private or global desktop to change the objects sharing mode. The desktop contains an object list containing the objects and their supported interfaces imported from the application by the interaction and collaboration DISCOVER servers. Every object encompasses the methods that allow monitoring the object status termed as views, and methods tat allow changing the parameters of the objects termed as commands. The portal has designated windows to display the application status messages, requested views results and command updates. The application status messages are divided into global status messages, global updates and local status messages. The global status messages indicate whether the application is in the computation or the interaction phase. The local status messages are used to indicate the status of the users request or command to the application. The portal provides tools like chat and whiteboard that provides the users with a means to share the information by text or graphic messages. The whiteboard has various drawing, writing and annotation tools. A whiteboard can be a user's private whiteboard or can be a part of a collaboration room. In the collaboration mode every users work is distinguished by the means of different color schemes and telepointers. The figure 5 shows the snapshot of the portal.

3.1.1 The client server communication model

The communication between the client and the server is completely asynchronous. This is attributed to the fact that the application alternates between the interaction and computation cycles. The client requests get queued at the server and are transferred to application sequentially during the interaction cycle. Due to the asynchronous nature of the application and the web based client server model, the client keeps polling the server to get the responses for the requests sent by the client. This is accomplished by running a



```
******** AMR Properties ********
Absolute MaxLev = 3
Current MaxLev = 3
Threshold = 0.005
RegridEvery = 4
BufferWidth = 2
BlockWidth = 1
MinEfficiency = 0.85
```

```
<vincent>refining might work
<vincent>
<vincent>
```

```
FineLevel : 1
Iteration# : 11
FineLevel : 2
```

```
vincent
volker
sriram
```

```
Status: INTERACTING
Status: INTERACTING
```

```
5. Your request: XSlice has been sent
```

Figure 5: - Snapshot of the DISCOVER portal

thread termed as the client update thread. The implementation of the portal allowed the user to be connected to only one application at a time per client. As the communication is asynchronous, keys have to be used to determine the corresponding client to the response received at the server end. These keys proved to be insufficient to map the responses to the correct application if the client was connected to more than one application. If the user needed to be connected to any other application at the same time, then a different instance of the client needed to be spawned. Due to this limitation, a collaborator would usually limit him to one application at a time.

3.1.2 Multiple application support

The main aim of DISCOVER is to provide application access to the user for monitor and control by placing the least amount of restrictions and providing the additional functionalities of collaboration by use of various tools. One of the major drawbacks of the portal was the lack of multiple application support. In order to access the portal for a purpose of chatting with another collaborator, one had to request an interaction with the application. Due to these restrictions imposed on the user, the portal was revised to counter them.

One can now access the portal even in absence of any active application and utilize the basic tools. When the user signs in to the portal, the list of applications and the list of users are presented to him. The user can now select to interact with any or all of the active applications from the list using the same portal. He is further presented with the option of holding individual chat sessions with any of the users without application interaction. The focus of the portal has now been shifted from interaction with the application to user collaboration. Whenever the user interacts with an application, a new panel is created for the application interaction within the portal. The look and feel of the new portal is still the similar to the earlier design to make transition simple. The figure 6 describes the new client server communication model.

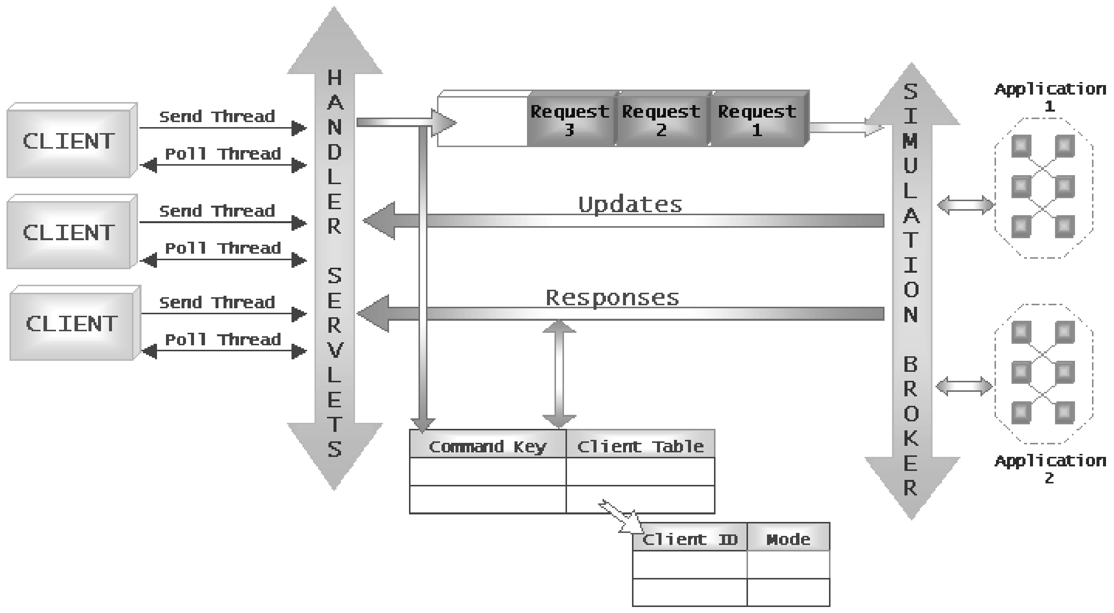
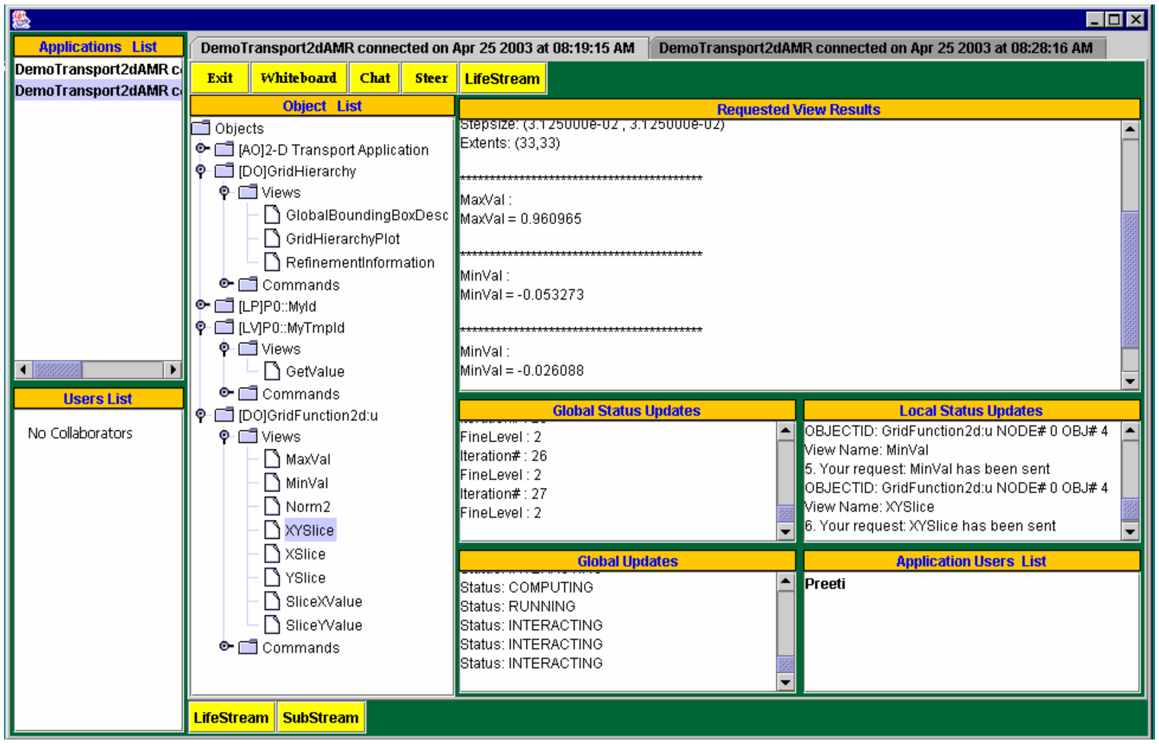


Figure 6: - Client Server Communication Model in DISCOVER

The portal now consists of multiple application interaction frames distinguished by the application identifiers received by the servers. As described earlier the application alternates between computation and interaction cycles. Hence the client has to maintain the send and the poll threads. Keys were needed earlier to distinguish the client to the response. But now as the client to application mapping has been changed multilevel keys have been added to facilitate proper response delivery. The server maintains a two level hash where every request or a command now has a unique key. The server queues the responses for every client in a client table depending on the mode of the client. As the client can have different collaboration modes for different applications, the keys are now a combination of the client identifiers and the application identifier. When the server receives a new response form the application for the request sent by the client, it appends the appropriate client-application queues based on the collaboration modes. The main goal of the client is to provide thin clients. Hence the least amount of complexity has been added to the clients to make the clients lighter to support easy web access. As every application interaction frame is now a new instance in the client portal, every instance now has its own send and poll threads. This helps keep the overall client structure simple as now another level or keys or hash tables need not be maintained in the client. As every instance has its own poll threads, the response times that are very important in real time monitoring systems, are kept very low. The figure 7 shows the view of the current portal.



```
Stepsize: (3.125000e-02 , 3.125000e-02)
Extents: (33,33)

*********************************************
MaxVal :
MaxVal = 0.960965

*********************************************
MinVal :
MinVal = -0.053273

*********************************************
MinVal :
MinVal = -0.026088
```

```
FineLevel : 2
Iteration# : 26
FineLevel : 2
Iteration# : 27
FineLevel : 2
```

```
OBJECTID: GridFunction2d:u NODE# 0 OBJ# 4
View Name: MinVal
5. Your request: MinVal has been sent
OBJECTID: GridFunction2d:u NODE# 0 OBJ# 4
View Name: XYSlice
6. Your request: XYSlice has been sent
```

```
Status: COMPUTING
Status: RUNNING
Status: INTERACTING
Status: INTERACTING
Status: INTERACTING
```

Figure 7: - Snapshot of DISCOVER portal with multiple Application Support

Chapter 4

Interaction Streams in DISCOVER

The Interaction is composed of documents organized by the events that have taken place. These time ordered events range from an application status update to a complex whiteboard session. As geographically scattered users create the interaction streams, the problem of unsynchronized clocks is dealt by considering the servers clock to be the standard. This time is displayed on the local portal on the users machine. The stream is further categorized into two kinds of streams, namely User Stream and Application Stream.

A User Stream contains the log of the private and group sessions that the user has had in his lifetime of usage of the portal. This is a very important feature for the client as it provides him with a complete visual log and session replay features of the workspaces.

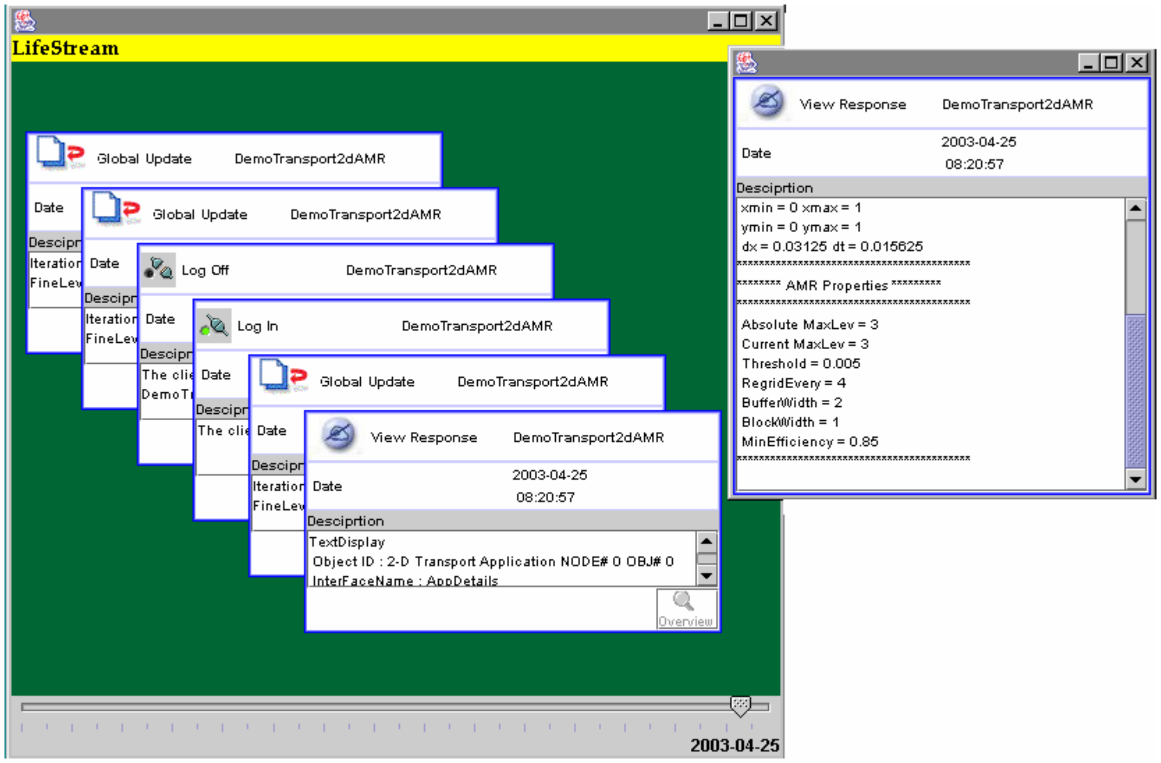
An Application Stream contains the complete information of the application’s life cycle, such as the status of the objects at particular time intervals, the users registered, the time the users logged in to interact with the application, the monitoring and steering requests. This stream enhances the security of the system as it provides a powerful medium to audit the application by the administrator of the application.

4.1 User Stream

User Stream is a complete log of the users work and interaction data in the DISCOVER portal. All the registered users of the DISCOVER system possess and own their personal streams. Our system provides the flexibility to the owner of the stream to decide the space for the storage of the stream. At present the DISCOVER server also provides storage in the pool of its resources, but as the streams are ever expanding, we expect the owners to have designated storage areas for their applications. A comprehensive log of all the clients and applications streams can be an issue of concern. But as this is the development stage the storage is considered to be vast less. This feature could be made optional in the further releases of the interaction streams. We extend the importance of this feature to the increase in the security of the system.

4.1.1 User Stream View

A user can request to the view of his Stream by the clicking the Lifestream button provided to him in the portal. The User Stream View displays a sequence of time-ordered documents ranging from present to the past. The date annotated scroll bar provided at the bottom of the user stream interface provides the capability of flipping through the stack of the documents. The user can bring a certain document in focus by selecting the document from the stack in the stream view. The document selected is rolled over to the front and hence the rest of the stack is refreshed with the appropriate information. Every document holds the information about the state of the application or the action taken by the user. These documents present concise information of the event, thus enabling the user to pick out the document of interest with ease. These documents hold various kinds of event information like user log in or log off, request of a view, global updates etc… This information is divided in to categories and icons have been associated with each category. These icons provide basic information like a simple user connect request to complex events like a response of a user request containing a plot graph. Simplicity is built into these documents by presenting concise information of the event to establish the required characteristics.


LifeStream
Global Update DemoTransport2dAMR
Date
Descipr
Iteration
FineLev
Global Update DemoTransport2dAMR
Date
Descipr
Iteration
FineLev
Log Off DemoTransport2dAMR
Date
Descipr
The clie
DemoTr
Log In DemoTransport2dAMR
Date
Descipr
The clie
Global Update DemoTransport2dAMR
Date
Descipr
Iteration
FineLev
View Response DemoTransport2dAMR
Overview
2003-04-25

Date 2003-04-25 08:20:57

Desciprtion

TextDisplay
Object ID : 2-D Transport Application NODE# 0 OBJ# 0
InterFaceName : AppDetails

View Response DemoTransport2dAMR

Date 2003-04-25 08:20:57

Desciprtion

```
xmin = 0 xmax = 1
ymin = 0 ymax = 1
dx = 0.03125 dt = 0.015625
******************************************
******** AMR Properties *********
******************************************
Absolute MaxLev = 3
Current MaxLev = 3
Threshold = 0.005
RegridEvery = 4
BufferWidth = 2
BlockWidth = 1
MinEfficiency = 0.85
******************************************
```

Figure 8: - Lifestream view in DISCOVER

This stack of pages is just a means to get to the desired event faster; hence the details of the event can be obtained by the click of the overview button provided at the bottom right corner of the document, which opens a new window to display the complete information of the event. Figure 8 displays such a user stream. The top label of the document depicts the kind of the event by the use of icons and the name of the application is depicted right next to it. The date and the time the document's creation is presented below this information. This is followed by the description of the event.

4.1.2 User-Application Stream View

The User Stream view provides complete log of the users interaction on the discover portal with all the applications. A user's interaction with an application at any time is an indication of the user's interest in the application. Hence we decided to include a user application stream view. This view is similar to the User stream in all aspects, but the stack now consists of the documents from the current application in focus. Selecting the Lifestream button provided in the application interaction panel can create this stream view.

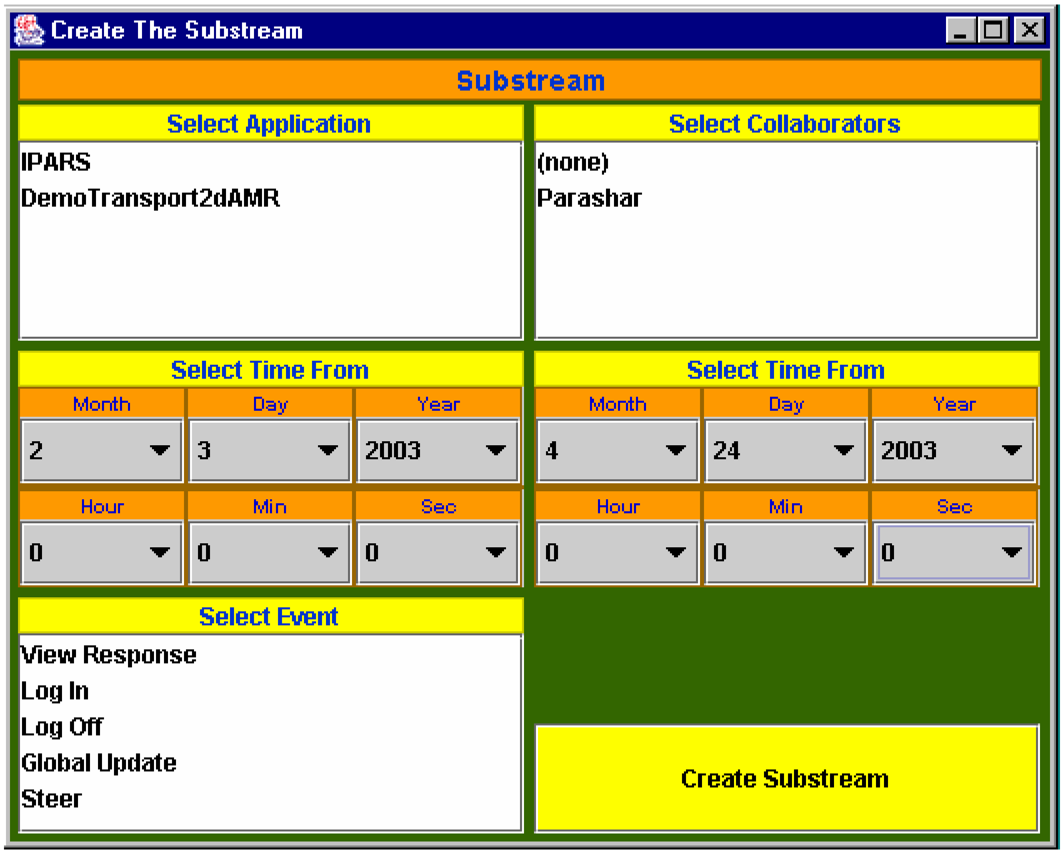
4.1.3 User Substream View

The User Stream View provides a complete log of the user's work and the user's application interaction in the DISCOVER system. Dealing with such huge amounts of information can be cumbersome when the user is looking for just a specific event. To deal with such needs of the user, a User Substream View has been added to the system. Figure 9 below represents such a view.

This view consists of 5 main elements.

1. Application List: - This is the list of the applications that the user is registered to in the DISCOVER system. This list is obtained from the DISCOVER server.
2. Collaborators List: - This list consists of the users who are registered to the same applications as the user and hence are termed as the collaborators.
3. Time From: - This allows the user to select the lower limit of the time duration of the user's interest.

4. Time To: - This allows the user to select the upper time limit of the duration of the user's interest.
5. Event List: - This list consists of the kinds of events. The events are divided into eight categories. All of these categories are contained in the Event List.


Create The Substream

Substream

Select Application	Select Collaborators
IPARS	(none)
DemoTransport2dAMR	Parashar

Select Time From

Month	Day	Year
2	3	2003
Hour	**Min**	**Sec**
0	0	0

Select Time From

Month	Day	Year
4	24	2003
Hour	**Min**	**Sec**
0	0	0

Select Event

View Response
Log In
Log Off
Global Update
Steer

Create Substream

Figure 9: - Substream View in DISCOVER

All the lists generated by accessing the information from the DISCOVER server's database resources. The user has no rights to edit or enter additional information into these lists. This prevents the user from unauthorized access into other User's stream. We have imposed this restriction to maintain the security imposed by system by assigning appropriate privilege levels for all the users for all registered applications.

Substream creation is probably the most important feature provided in the interaction tool. Selecting any, some or the entire criterion, one can now obtain a stream of filtered documents. Multiple values can be selected from any of the lists to comprise the decisive factors for the Stream. The Create Substream Button when clicked gathers the selected conditions and creates a union of all the conditions. As the streams are created on the fly, the stack created is dynamic and no permanent storage is assigned to it.

All the elements except the list of the collaborators create the substream from the user's personal stream. The condition of the query is a result of the combination of all the conditions specified by the user in the substream view. All the documents fulfilling these conditions now comprise the new substream. These documents are displayed in similar user interfaces as the Users Stream frame. The Collaborators List is of special interest here as this now allows the user to index into stream owned by his collaborators. The substream created is an intersection not a union of the two streams, hence the events occurred during the collaboration of selected collaborators and the user forms a collection or a group of these documents. Even when two people are in collaboration mode, their privilege levels can limit the information that is displayed to the user in his portal e.g. a user with a privilege level lower than 2 cannot request views of objects of the application and also cannot obtain the views requested by his collaborator. Due to the synchronous nature of the DISCOVER system, the users need to be in collaboration mode in order to obtain any information about another user. This feature has been incorporated to build the security of the system.

4.2 Application Stream

The main purpose of the Application Stream is to provide the owner of the application auditing capabilities. Every application is registered to the DISCOVER server by a user termed as the owner of the application. This auditing stream is accessible only to the owner to the application. Similar to the User Stream, this is stored in the DISCOVER system's pool of resources. This feature could be made optional in the further releases of the interaction streams. We extend the importance of this feature to the increase in the security of the system.

4.2.1 Application Stream View

The Application Stream View is similar to the User Stream View. The application Stream view consists of a stack of time-ordered documents. Every document holds the information about the state of the application ranging from application connects to application disconnects. The date annotated scroll bar provided at the bottom of the stack provides the capability to the user to flip through this stack of documents. These pages just provide minimal concise information of the event. This stack of pages is just a means to get to the desired event faster, hence the complete information of the event can be obtained by selecting the document, which opens a new window to display the information.

The Application Stream can be used to insert some asynchronous characteristics into DISCOVER system. In the current system, the state of the application can be monitored only when the user is connected and interacting with the application via the portal. As the Application Stream logs all the updates on the status of all the objects present in the application, one can access these logs and get the required information. But the main motive for the application stream is to serve as a basis for an auditing stream.

Chapter 5

Design of Interaction Streams in DISCOVER

This chapter presents the design of the new tool called the Interaction Streams in DISCOVER.

5.1 Elements comprising the User Stream

The User's Stream is comprised of the following pieces of information.

- Application Identifier: - This is the identifier assigned by the server that is assigned to the application when the application is first initiated on the server.
- Time: - This is a combination of the date and the time when the event was initiated. This is the primary key of the stream. As the server sends the requests sequentially to the client, all the events have different time stamps associated with them. The server puts the timestamps when any information is sent to the client. Thus in case the users are in collaboration mode, the responses or updates received at the users end have the same time stamp. At the time of the substream formation, it is this primary key that differentiates whether the event was a collaboration event.
- Description: - This field contains all the information about the event. It can contain information ranging from the chat message to the plot points of the graph of a View Response. The complete description is used only when the document is selected for complete view.

- Event: - This field consists of the kind of event that comprises the document.

5.1.1 Classification of Events in The User Workspace

As explained in the earlier chapter the documents contain information based of different kinds of interaction events in the users portal workspace. A user's complete workspace is a combination of the workspaces of all the applications that the user is currently interacting with. The events are categorized into 2 kinds based on their source of origin. They can be either user initiated or application initiated. User initiated events can be personal or a can be sent by a collaborator when the user is in collaboration mode. The application-initiated events usually consist of the updates. These events are listed below.

	Event Name	Description	Event Type	Symbol
1	Client Log In	The event when the user sends and interaction request to the application.	User Initiated	🔌
2	View Response	The event when the user requests a view. The response of the view is stored as View Response. View response can be either textual or graphical.	User Initiated	◉
3	Steer	The event where the user sends a steer request	User Initiated	☸

4	Global Update	The event where the application sends the global update messages to the users portal	Application Initiated	
5	Client Log Off	The event where the client sends a disconnect request to the application	User Initiated	
5	Application Connect	The event where the application is started on the DISCOVER server	Application initiated	
6	Application Disconnect	The event where the application is complete or disconnects due to some problem	Application initiated	
7	Chat Event	The event where a user sends a conversation message to another user.	User initiated	
6	Whiteboard Event	The event where a user draws or writes on the whiteboard	User initiated	

Table 1:- Categories of Interactions

5.2 Communication Model with the Interaction Stream Tool

The incorporation of the Interaction Stream tool has introduced changes in the system architecture of DISCOVER. This section describes the changes in the design of the entities of the system.

5.2.1 System Architecture of DISCOVER

The figure 10 shows the system architecture of DISCOVER with the introduction of the Interaction Stream Interface. The design now consists of the following elements.

- P2P network of DISCOVER servers

  The DISCOVER servers form a p2p network providing the necessary services and the messaging substrate. This network is the key to providing seamless interaction between the applications and the clients. The applications connect to any of the available resources in this p2p network. Services have been introduced to maintain consistent logging of the user and application information in the storage pool of resources. The primary reason to introduce this service at the server is to maintain thin clients and moreover all the client events are processed through the server. Hence it makes this integration meaningful.

- Storage resource pool

  This storage pool comprises of object relational distributed databases. These resources provide resource access services to the other entities in the network. The servers use these services to store or retrieve information from the resource pool. These services also play a great role in the incorporation of the new tool of interaction streams at the client.

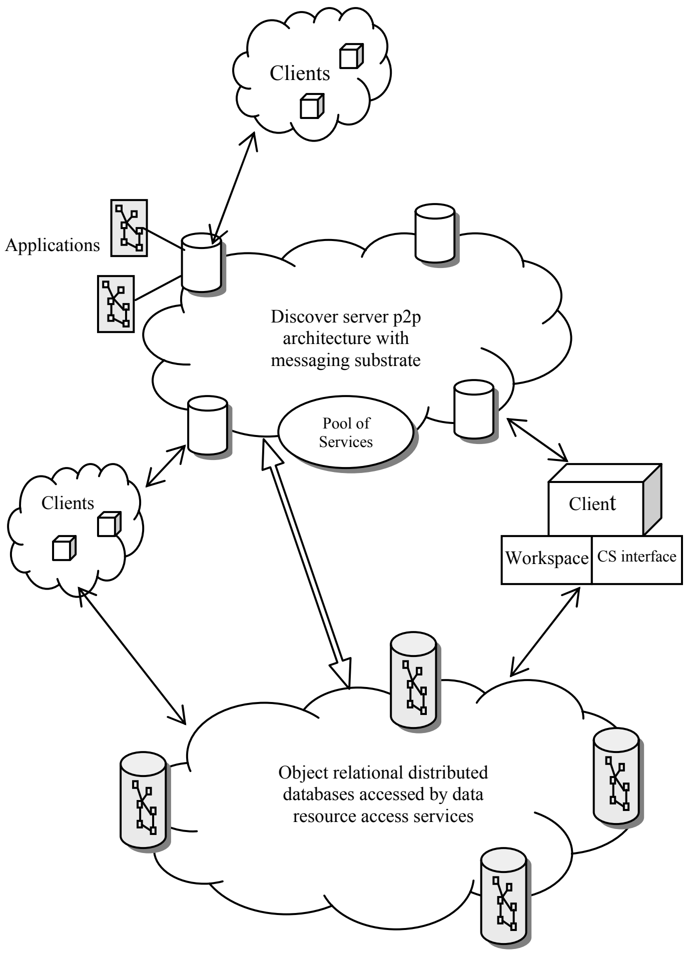

Clients
Applications
Discover server p2p architecture with messaging substrate
Pool of Services
Clients
Client
Workspace
CS interface
Object relational distributed databases accessed by data resource access services

Figure 10: - The system architecture of DISCOVER

- Client

The thin client view is the current portal that supports multiple application interaction. Every active application interaction comprises the workspace of the client. The client utilizes the services provided by the messaging substrate in order to interact with the applications. An interface called the Interaction Stream Interface has been added to the client. The responsibility of this interface is to access the services provided by the resources pool and visualize the streams. When the interaction streams tool is activated, this interface develops a query based on the conditions declared by the client and then requests the interaction stream service for the set of the documents that fulfill the presented user specified condition. Once the client receives this set of document, the interface provides it meaning by presenting the user with a visual representation of these documents. The user can now flip through the stack of the documents as discussed earlier.

5.2.2 Client-Server Communication for the Formation of Interaction Streams

The figure 11 depicts the new client server communication model for user-initiated events. The earlier sequence of events persists in the communication. Steps have been added in the communication to enable application and client event logging. Once the response or any interaction update is received from the application, it is assigned a timestamp at the server. These queued responses are matched with the keys and passed on to the poll threads of the client. Once these requests are sent to the threads, the events are logged in the server's storage resources in the user's interaction stream. Thus in case of a collaboration event, the timestamp of the event is the same for all the users in the collaboration mode. Even though the responses are sent sequentially and arrive at different times from the server to the clients, it still possesses the unique timestamp.
Hence we have assigned time as the primary key in the interaction stream.

Due to the client server nature of the portal, to get application-initiated events like a thread running at the portal handles the global updates that send request to receive the updates. Hence even though the event seems to be initiated by the portal, it is an application event that needs to be communicated to the portal. Hence all the events come as responses to the portal.

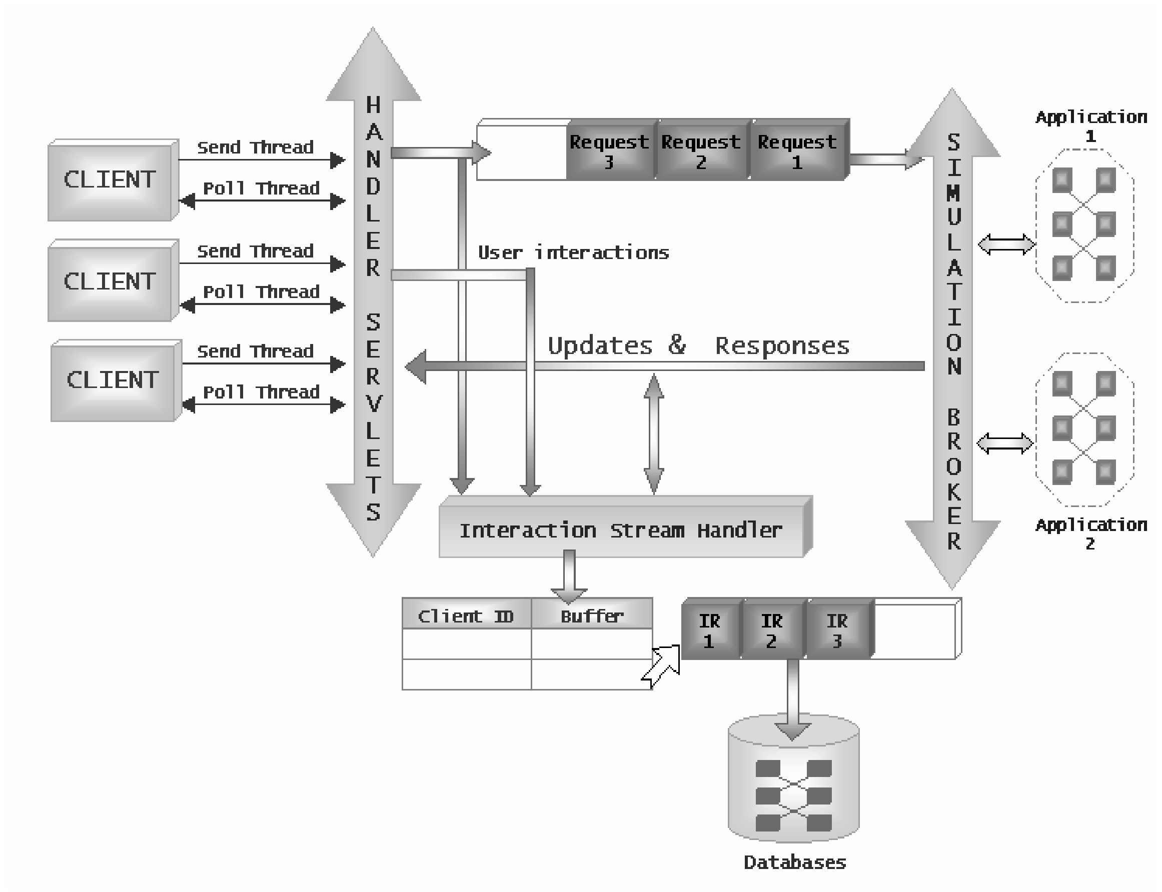


Figure 11: - Client Server communication with Collaboration stream module

Due to the client server nature of the portal, to get application-initiated events like a thread running at the portal handles the global updates that send request to receive the updates. Hence even though the event seems to be initiated by the portal, it is an application event that needs to be communicated to the portal. Hence all the events come as responses to the portal.

5.3 Issues faced in the design of the tool

For a smooth operation of such a tool, the following issues had to be considered and dealt with. Because the tool requires dealing with massive amounts of information and repeated access of the databases in the resource pool, response time and memory overrun are the major concerns.

5.3.1 Stream logging

As explained earlier, the server is responsible for logging all the events into the stream. This translates to repeated insertions into the databases. Latencies enter into the system from various sources. Due to the client server model of the system, the client request is sent out and is queued at the server until it can be processed. Hence there is latency in the request- response from the client-server. The application responds to the user request sent to the server only when it enters the interaction phase. This is another cause for the latency being built in. Moreover these responses are further queued at the server with other updates like user list updates, global updates etc. Hence these responses arriving at the client sequentially contribute to the total latency. In order to make the interaction and collaboration real time, latency should be kept under check.

We have taken a buffering approach to counter the problem of the latency. As discussed earlier, the server thread maintains a queue of the responses for the clients in a hash indexed by the client id. The responses are sequentially sent to the client and a copy of the response is stored in a buffer. When the buffer is full or the client decides to terminate the interaction with the application, the buffer is emptied into the database. As the time taken to connect to a database is much more significant than the time taken to enter records into an open database, the buffer scheme helps in controlling some lag introduced by the logging of events. The size of the buffer can be decided by considering factors such as the kind of database used, namely relational or object oriented, local or remote.

We consider the databases to have unlimited memory right now. But in the future versions of the system, we can introduce concepts of deleting records from a user's personal workspace or archiving old records.

5.3.2 Stream Retrievals

As the interaction stream is set of all the events occurred during the lifespan of the client with all registered applications, the record count is limitless. These large records pose a threat of memory over-run. This factor possesses a lot of importance as it threatens to run down the system. Moreover the numbers of records accessed also translate directly into latencies.

A scheme called zoom in is created to counter this problem. The stream views contain the stack of documents arranged from the present to the past in the view. These documents can be flipped through by the use of the date annotated scroll bar. This stack of documents is created by the query to the database based on the user enlisted conditions. The stream interface retrieves records arranged going from present to past and adds a condition limiting the number of records extracted from the database. This is under the assumption that the user is interested in the current information more than the past. But once the user scrolls towards the end of the active record set, the stream view zooms in to the past. At a certain point there is only one active set of records. Hence new records fulfilling the same condition but ranging in time from the lower limit of the current set of active records and to further in the past are extracted. Some over lap is maintained between the current and the old working sets of the records. Similarly when the scroll bar is moved to a location in time currently higher than the active set, the new set (earlier set) is reloaded.

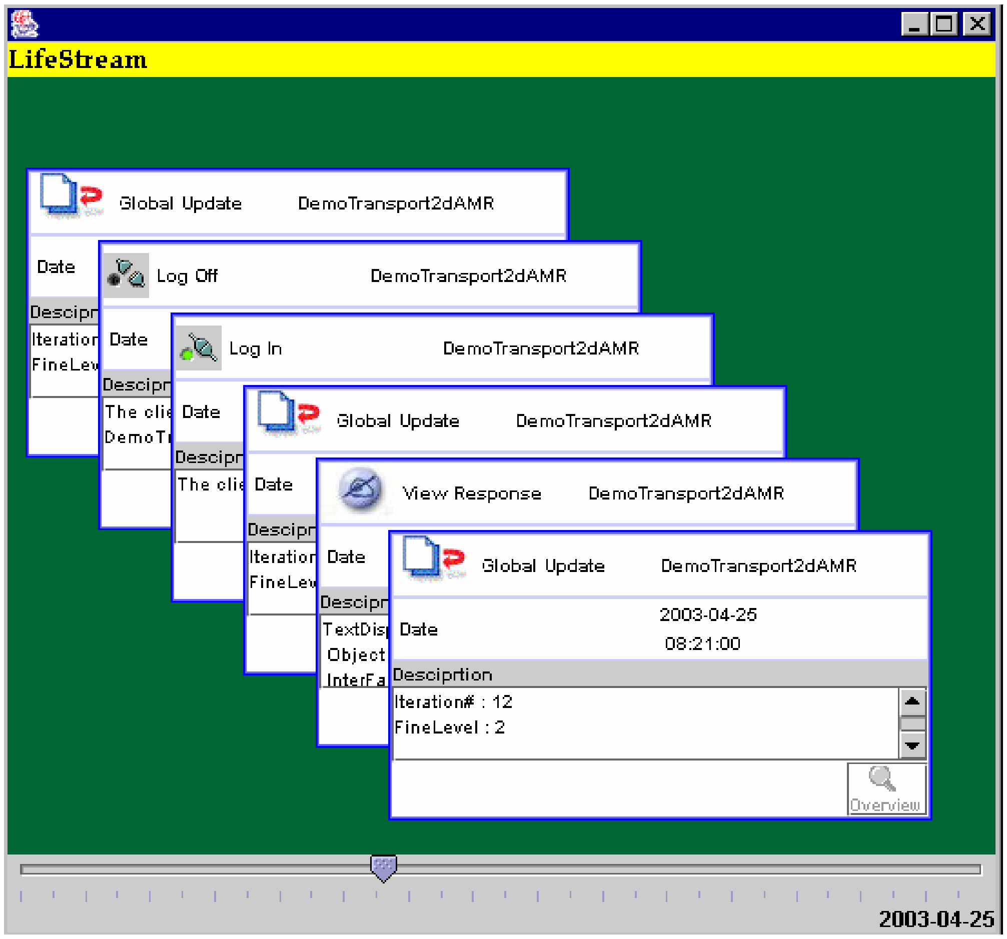


Figure 12a: - The view of the lifestream before the activation of zoom in feature

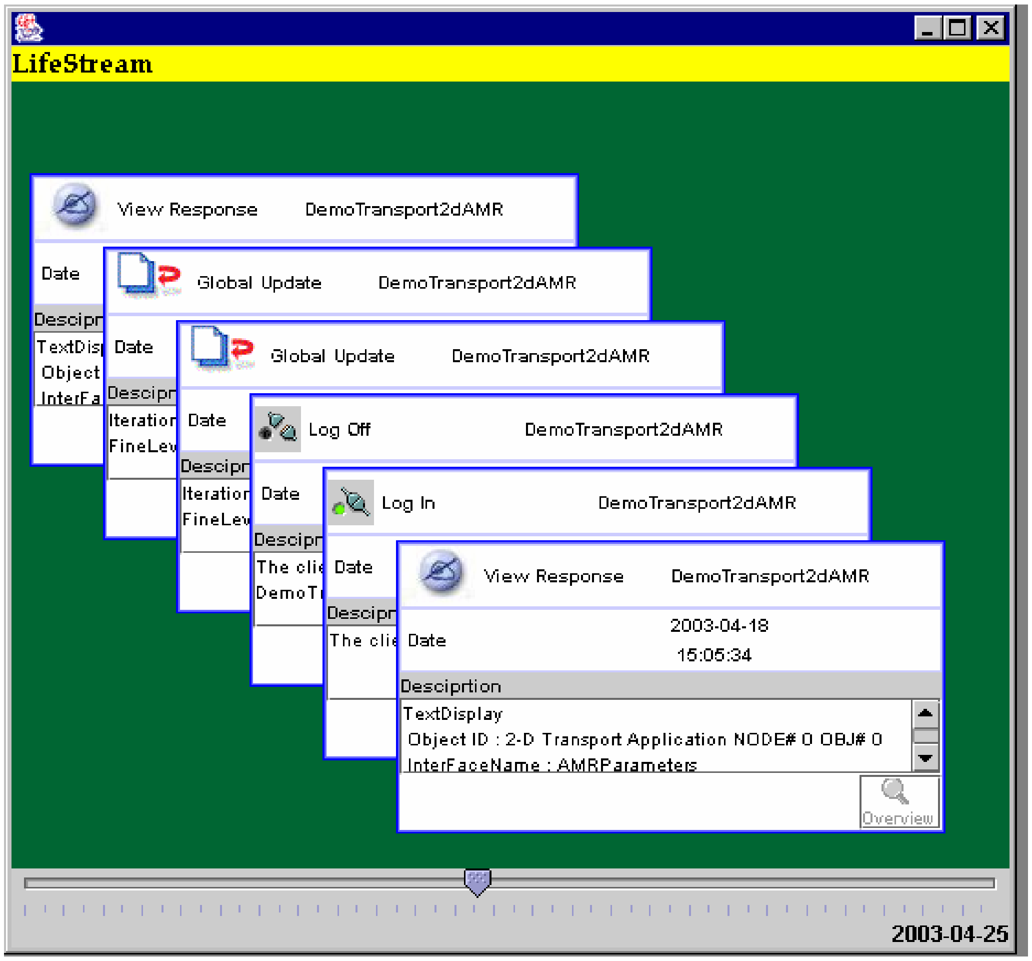

LifeStream
View Response DemoTransport2dAMR
Global Update DemoTransport2dAMR
Global Update DemoTransport2dAMR
Log Off DemoTransport2dAMR
Log In DemoTransport2dAMR
View Response DemoTransport2dAMR
Date
2003-04-18
15:05:34
Desciprtion
TextDisplay
Object ID : 2-D Transport Application NODE# 0 OBJ# 0
InterFaceName : AMRParameters
Overview
2003-04-25

Figure 12b : - The stream view after the activation of the zoom in feature

This feature is an attempt to keep in check the memory and time constraints imposed by the nature of the tool. The figure 12 depicts the tool. Figure 12a shows the initial stream view of the user. Figure 12b shows the activation of the zoom in feature. The scroll bar is now extended to incorporate more records. But only one working set of records is maintained. The overlap in the two sets of records maintained here are 50.

Chapter 6

Implementation and Evaluation of Interaction Streams in DISCOVER

6.1 Implementation

The interaction stream tool has been designed to be used with DISCOVER. Hence to incorporate the tool into the working model of DISCOVER, the established framework and implementation environments have been used. The current framework of the system has been described in chapter 3.

6.1.1 Stream Retrieval Implementation

This section describes the mechanisms and the interfaces that have been built in the client to generate the streams.

- **Interaction Stream Interface**

  The Client to stream communication uses a channel denoted as the stream channel. This is an independent channel established between the client and the pool of resources. As described earlier in chapter 5, the client implements a *InteractionStreamInterface*. This interface includes methods for authentication of the client to the database resources. This interface provides the only means to the client access the personal stream. This interface also includes methods to generate queries for the conditions stated, querying the databases to generate the streams. This interface is further inherited by two modules namely substreams and User Streams.

The substreams interface deals with generating the complex conditions for the user query provided by the values of the keys selected from the substreams user interface. Database access support has also been provided at the client side. These requests are not sent via the server because the primary purpose of the server is to maintain interaction between the client and the application. This task does not fall into that category because it is a review of the work done. A stream of the review work is not created even though the streams are accessed via the client portal.

- **Stream User Interfaces**

Using java swing creates the user interfaces. We chose this in order to allow easy resizing of the application windows. These user interfaces contain methods to convert the textual data into visual representation. Some other methods comprise of the enabling of scrolling through the stream, selection and complete description of particular record, zoom in to the past and present. The Interaction Stream UI and SubStream UI inherit these stream user interfaces.

6.1.2 Stream Logging Implementation

This section describes the modules created to enable logging of the events to create the stream. As discussed earlier, the server is comprised of servlets performing their designated tasks. The interaction and steering servlet is responsible for handling all the requests and commands sent by the user. There is yet another servlet employed to take care of the interaction needs. An interface called the stream interface has been created to make this task possible. This stream interface is inherited by the servlets. The interface consists of methods to insert the records into the database. This interface also consists of methods to enable authentication of the client to the databases in order to access the correct streams. The database handler built in the server makes the database communication possible. This interface provides other methods such as method to add records to the buffer, method to insert all the records from the buffer into the database for delayed group insertion.

6.1.3 Database design

The original database resources are built on the relational database MYSQL. Hence we decided to carry create the streams in this database. The figure 13 presented below, lists the tables that are present in the databases.

The tables comprising the databases are:-

1. Users

   This table contains the user information such as the name of the user, user id, and password. This table is used to authenticate the user when the user logs in to the system. The status filed indicated if the user is currently available on the portal or not.

2. ACL

   This purpose of this table is to authenticate if the user is registered for the application and check the privilege level of the user for the particular application. It contains fields UserID , Application Name, Privilege and Status.

3. Applications

   The records of this table contain the applications that are currently running on the discover server. Hence when the user logs in to the system, a query on this table and on acl is made and the list of active applications for which the user is registered is passed to the portal. The various fields comprising the table are Application Name, Application ID (assigned by server), Server ID and Status.

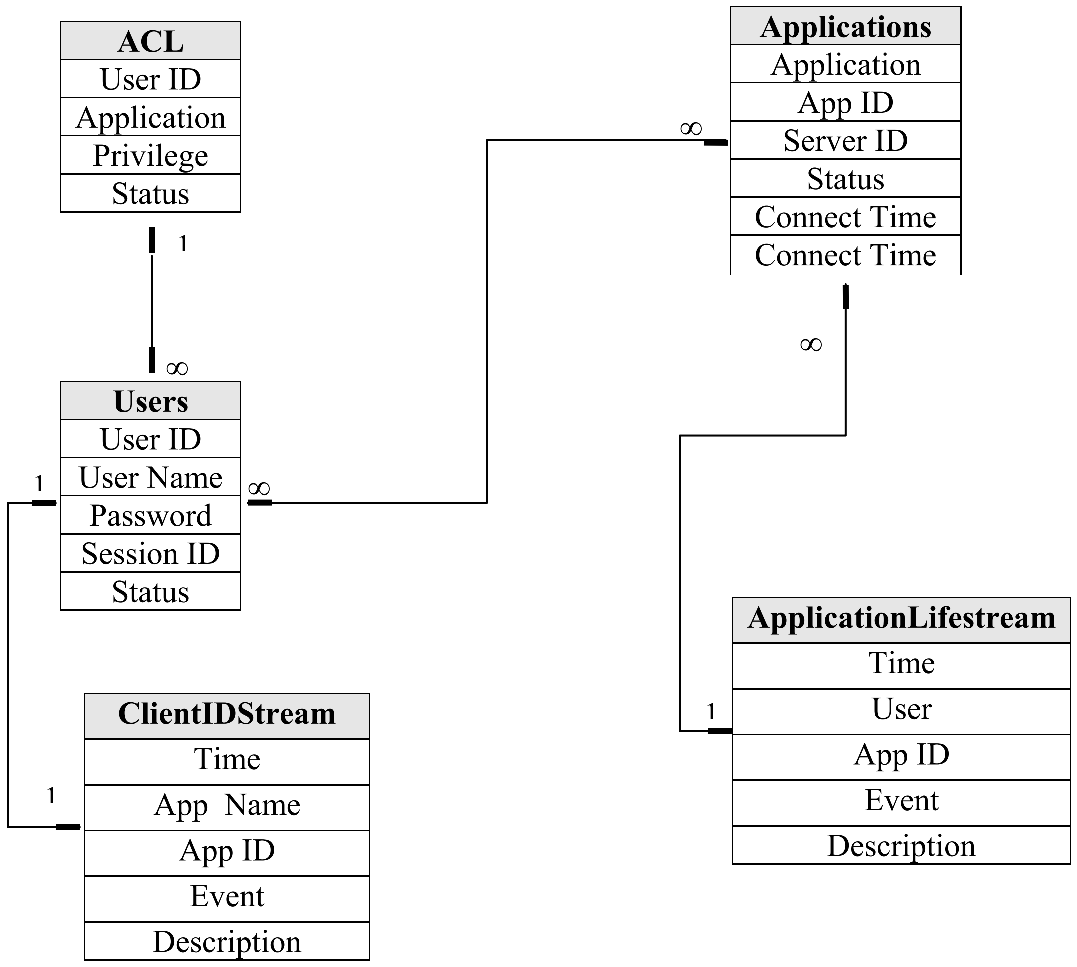


Figure 13 – Structure of Database

4. ClientIDStream

Every user of the system has his own Stream. The name of the table is a combination of the userID and Stream. It contains the logs that comprise the stream. The fields include, Time – the primary key, Application Name, Application ID , event – these are divided into eight categories and Description – containing the description of the event.

5. ApplicationStream

   This table contains the stream of the application. Access to this table is provided only to the owner of the application.

6.1.4 Applications Used

DISCOVER system has been used to provide interaction capabilities to a number of applications. The tools developed have been tested by the running some of these applications on the framework. A list of the applications used for this purpose are presented below

1. Reservoir Simulation - GrACE[9] has been integrated with IPARS (Integrated Parallel Accurate Reservoir Simulator) [[26], [10]] to provide a problem solving environment for parallel adaptive porous media and reservoir simulations. This tool is developed to model the behavior of fluids in permeable geologic formations such as petroleum and natural gas reservoirs and ground water aquifers. Deployment and management of IPARS instances uses services provided by DISCOVER [10] and Globus[4].

6.2 Evaluation

Experiments were conducted to test the overheads of incorporating Interactions streams into the DISCOVER system. The experiments ranged from calculating the overheads of time and memory to form and retrieve the streams.

6.2.1 Experiment 1

This experiment aims at quantifying the overheads introduced by the addition of the logging interactions at the server end. The experiment consisted of calculating the time required to empty the buffer into the database. The overheads for this experiment are plotted in the graph in figure 4. The peaks in the graph represent the time taken to perform the group logging from the buffer. There are intervals of time where no record is logged to the database. An overhead of 6.05ms on an average is introduced per interaction at the server.

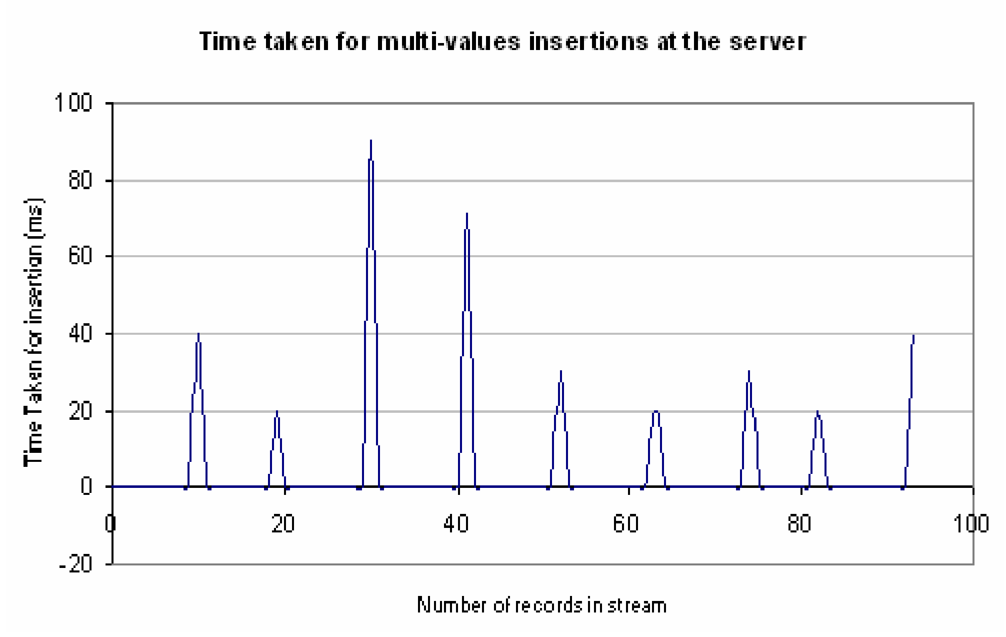


Figure 14:- Graph depicting time overhead incurred for Interaction Streams

6.2.2 Experiment 2

This experiment calculates the memory used for the storage of records in the interaction streams. The graph in Figure 5 shows a linear increase in the memory used with the increase in the number of records. On an average a record consumes 220 bytes of memory. As the entire information can be contained in such small records, the streams can store a large number of records before it runs out of memory.

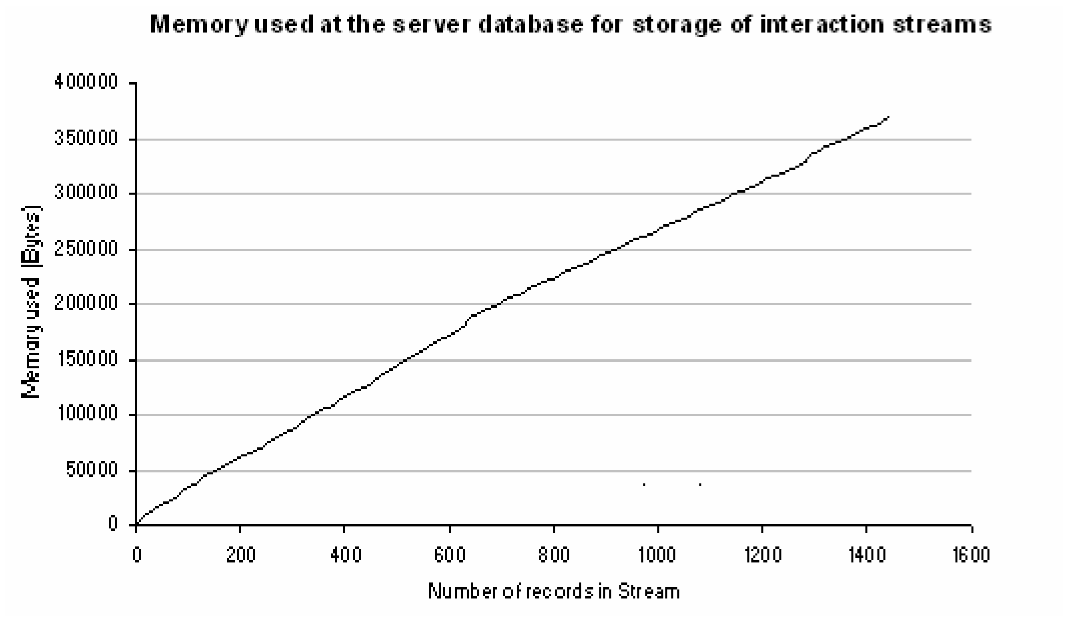


Figure 15:-. Graph depicting Memory usage for storage of Streams

6.2.3 Experiment 3

This experiment is used to estimate the time overheads in the representation of the Interaction Streams. The experiment consisted of calculating the time required to load the active set of records from a user interaction stream fulfilling the conditions. The values of access times are plotted against the total number on records contained in the stream. The values from the graph in Figure 6 show that the access times of these records are limited in a time range based on the sizes of the records retrieved. These times do not increase with increase size of the stream because only a subset of the information is retrieved at any give point.

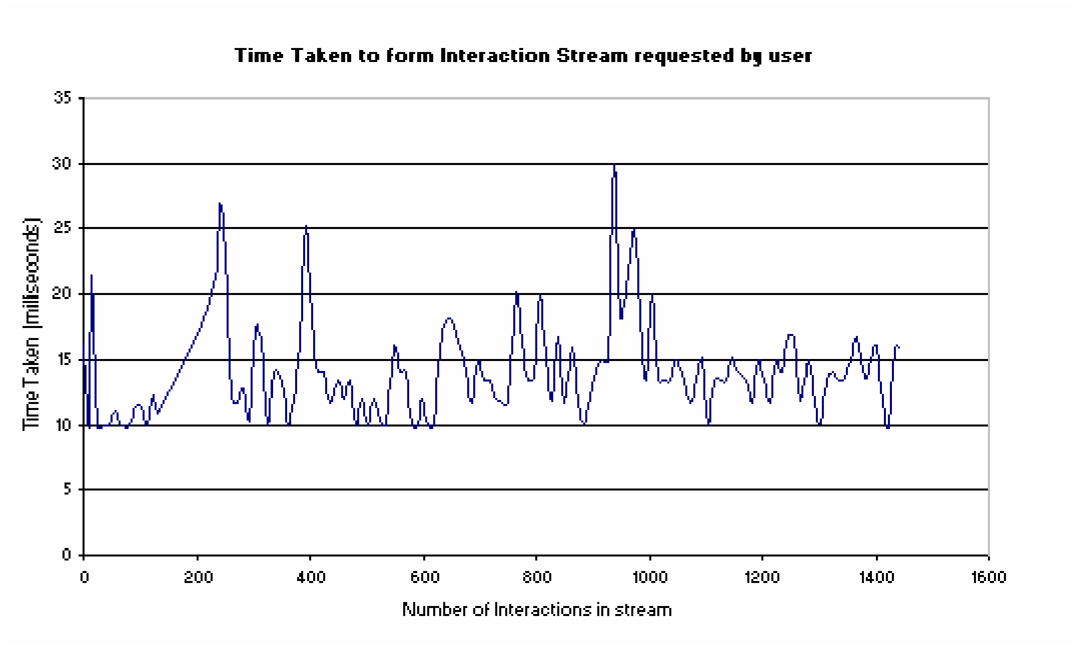


Figure 16:- Graph depicting time overhead incurred for extraction of records from Interaction Streams

Chapter 7

Conclusion and Future Work

The system presented here is an attempt to increase the usability of the DISCOVER. The provision of multi-application support and the integration of a visualization tool called Collaboration Streams are they key contributions of the thesis. The designed tools are based on the requirements gathered and issues faced from the collaborators working groups in the current DISCOVER system. The key challenges faced were to increase the usability of the system by providing session logging and retrieval support and visual representation of the data gathered.

This thesis presented the design and implementation of a tool used for session logging and retrieval that aids in the visualization and review of the collaborative environment. It further explains the incorporation of this tool into DISCOVER – a web based computational collaboratory. It further explains the solutions to the challenges faced, like data storage, data retrieval, data representation, scalability, and privilege checks, ease of use and memory usage. The client supporting multiple application interaction has been designed and developed. These contributions have contributed towards making the system more usable.

The revised client with the tool and the multi-application support is now available and provides the scientists with the ease of being connected to multiple applications and the capability to review their work in the collaboratory with the help of the intensive logging and retrieval resources built into the system.

This was achieved by the development of the Interaction Streams interface. This interface encapsulates methods for authentication of the client to the database resources and provides a means for the client to access his personal stream. It also provides methods to generate queries for the conditions stated, querying the databases to generate the streams.

One of the most important issues arising from the development of this tool is data storage. An immense amount of data is produced as a result of the stream logging, the data resources would saturate. Efforts need to be put in to counter this limitation. Techniques like selective data logging can be put into place. The logging can be made more intelligent by identifying each object and supporting querying on the values of these objects. A study can also be performed to measure the usability of the tool in the system and the issues faced by the usage of this tool.

We hope that the future work will address the issues faced and help build an efficient easy to use scientific collaboratory.

References

[1] DISCOVER (Distributed Interactive Steering and Collaborative Visualization EnviRonment), http://www.discoverportal.org.

[2] "DISCOVER: An Environment for Web-based Interaction and Steering of High-Performance Scientific Applications," V. Mann, V. Matossian, R. Muralidhar, and M. Parashar, *Concurrency and Computation: Practice and Experience*, John Wiley and Sons, Vol. 13, Issue 8-9, pp 737 – 754, 2001.

[3] Fertig, S., Freeman, E., and Gelernter, D. Lifestreams: An Alternative to the Desktop Metaphor. In *ACM SIGCHI Conference on Human Factors in Computing Systems Conference Companion (CHI'96),* pp. 410 - 411, ACM Press, 1996.

[4] Eric T. Freeman and Scott J. Fertig. Lifestreams: Organizing Your Electronic Life. *AAAI Fall Symposium; AI Applications in Knowledge Navigation and Retrieval*, 1995.

[5] Freeman, E. and Gelernter, D. Lifestreams: A Storage Model for Personal Data, *ACM SIGMOD Bulletin,* (March, 1996).

[6] Matthew Bianco, *An Interface for the Visualization and Manipulation of Asynchronous Collaborative Work within the DISCIPLE System*, Department of Electrical and Computer Engineering, Rutgers University, January 2000.

[7] "DARPA Project Summary: The DISCIPLE System, Rutgers University" 1997 http://www.caip.rutgers.edu/disciple/

[8] Gelernter, David. "ScopeWare – Information Management Systems" Mirror Worlds Technologies 1998. http://www.mirrorworlds.com/horizons/index.html

[9] Manish Parashar. Grid Adaptive Computational Engine. http://www.caip.rutgers.edu/TASSL/Projects/GrACE.

[10] Rajeev Muralidhar - *A Distributed object framework for the interactive steering of high performance applications.* Department of Electrical and Computer Engineering, Rutgers University, October 2000.

[11] Vijay Mann - *Middleware Architecture for Integrated Computational Collaboratories.* Department of Electrical and Computer Engineering, Rutgers University, October 2001.

[12] Java Servlet API Specification, http://java.sun.com/products/servlet/2.2/.

[13] Usability First – Online guide to usability resources http://usabilityfirst.com

[14] Apache Web Server, http://httpd.apache.org

[15] Apache Jserv Servlet Engine, http://java.apache.org

[16] Groove Networks : http://www.groove.net

[17] NetMeeting http://www.microsoft.com/netmeeting.

[18] S. Subramanian, G.R. Malan, H.S. Shim, J.H.Lee, P. Knoop, T. Weymouth, F. Jahanian,A. Prakash, and J. Hardin, .The UARC web-based collaboratory: Software architecture and experiences., *IEEE Internet Computing*, Vol.3, No.2, pp.46-54, 1999. See also: http://intel.si.umich.edu/sparc/.

[19] Cactus Computational Collaboratory. http://www.cactuscode.org.

[20] M. Russell, G. Allen, G. Daues, I. Foster, T. Goodale, E. Seidel, J.Novotny, J. Shalf, W. Suen, and G. von Laszewski, .The Astrophysics Simulation Simulation Collaboratory: A Science Portal Enabling Community Software Development.. *Proceedings of Tenth IEEE International Symposium on High Performance Distributed Computing*, August 2001.