2001

Investigation of real-time multi-user application development frameworks.

Harvinder S. Minhas
University of Windsor

INVESTIGATION OF REAL-TIME MULTI-USER APPLICATION DEVELOPMENT FRAMEWORKS

by

Harvinder S. Minhas

A thesis submitted to the Faculty of Graduate Studies and Research through the School of Computer Science in partial fulfillment of the requirements for the Master of Science degree.

University of Windsor

Windsor, Ontario, Canada

2000

0-612-62255-X

ABSTRACT

The Internet provides global connectivity and with it presents an enormous potential for supporting real-time collaborative-work (RTCW) applications. However, developing such applications from scratch is a complex and time-consuming task.

In this thesis, we investigate the use of frameworks to develop such applications. In particular we look at the Habanero framework that is designed for developing real-time collaborative applications. We also discuss the Internet from a real-time communication perspective since the ultimate success of RTCW applications would depend on the quality of service provided by the Internet to the data that flows through it.

ACKNOWLEDGMENTS

I would like to express my deepest thanks and appreciation to Dr. Richard Frost who as my supervisor guided and motivated me through out this thesis work. I would also like to thank Dr. Subir Bandyopadhyay and Dr. Cyril G. Rodrigues for their valuable feedback.

I would also like to thank all the faculty members of School of Computer Science for their support and encouragement through out my study period at University of Windsor. Also many thanks to the wonderful secretaries Mary and Gloria for their help.

Last but not the least I would like to thank all my friends for making my stay at University of Windsor memorable.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Problem Specification

Computer and network technologies have advanced tremendously over the last few years. The potential for real-time computer-assisted collaborative work over the Internet is huge. The benefits of such work are great – for example imagine auto-designers from various companies being able to work together on a new emission-control system over the Internet.

However, this potential cannot be fully realized at present owing to the lack of tools and infrastructure to assist the development of real-time collaborative work (RTCW) applications.

1.2 Thesis

The thesis is that the development of real-time collaboration applications can be facilitated through the use of frameworks.

1.3 Work Done

The thesis work involved analysis of technologies, which intend to provide tools and infrastructure to support the development of RTCW applications.

In particular, the work involved a critical analysis of one of the most advanced, yet still immature, software frameworks called Habanero. Habanero is an object-oriented framework that assists in the development of RTCW applications. The critical analysis of Habanero also involved the development of a collaborative text editor using this framework.

1.4 Summary of Conclusions

The critical analysis of Habanero showed that:

1. Habanero uses client/server architecture along with the concept of replication to achieve collaboration. The information, in the form of data objects, is replicated and sent to every interested participant. The server opens a TCP/IP socket connection to each client to pass on this information. This technique of opening a connection to each client would not scale well when large number of clients are connected.

2. There is a lack of complete documentation about Habanero API. This presents a serious impediment in developing quality collaborative applications.

3. The Habanero framework is developed using Java and hence it only supports development of applications that use Java as the programming language. This seriously limits the scope of the framework. The framework should be integrated using technologies like XML and CORBA to interface the Habanero applications with external systems or applications.

4. Habanero should be extended to take advantage of the new service models that are being proposed for the Internet. Currently, the Internet only supports the best-effort service model whereby no guarantees about the quality of service are made by the network. The new service models being proposed are the fair service, controlled-load service, predictive delay service and the guaranteed service models. These models are discussed in detail in chapter two.

5. The Habanero framework closely represents the black-box architecture. Hence, the developers don't need to know the insides of the framework in order to plug-in their applications into the Habanero environment.

6. Habanero encourages code as well as design reuse. This results in applications that are relatively easier to develop and maintain.

7. Habanero is a step in the right direction for achieving true collaboration. It is an excellent framework, which enables developers to build real-time multi-user applications in relatively little time.

1.5 Significance of this work

Developing real-time collaborative applications is a very complicated task since it involves programming at a very low-level. The Habanero provides us with a sophisticated framework that claims to facilitate the development of Internet-based RTCW applications. The RTCW applications we believe will be the next generation Internet-based applications and frameworks like Habanero will play big role in development of these applications. However, there is very little work done to investigate such frameworks.

This work carries out a critical analysis of Habanero.

The findings of this work demonstrate the need for Habanero to be made more scalable and better documented.

The work has also illustrated the need in general for tools like Habanero.

The work also investigates the present and future of the Internet, in terms of supporting real-time communication.

The analysis provided in this work can be used as a yardstick for evaluating any future similar frameworks

1.6 Structure of the thesis

The thesis is structured as follows:

1. The Internet from a real-time communication perspective.
2. An overview of Computer-Supported Collaborative Work (CSCW) with emphasis on groupware.
3. Investigation of different techniques to build RTCW applications.
4. Framework for Integrated Synchronous And Asynchronous Collaboration (ISAAC).
5. Detail discussion of Habanero.
6. The development of collaborative text editor using Habanero.
7. Statement of facts.
8. Conclusions.

Chapter 2

THE INTERNET FROM A REAL-TIME COMMUNICATIONS PERSPECTIVE

The Internet was initially designed to deliver low-bandwidth static-information for the scientific community. However, in the past few years, the Internet's popularity has exploded and it has become a global phenomenon. The Internet's simplicity and ubiquitousness has stirred tendencies among different communities to use the Internet for their own purposes. One of the ideal uses of the Internet would be to bridge the geographical distance between people without losing any interactivity that people experience when they communicate face-to-face. However, in order to achieve this level of interaction it is imperative that the Internet supports real-time communication.

This chapter takes a look at the Internet from the real-time communication perspective.

2.1 Real-time Communication: An Overview

Real-time communication is an "exchange of time-sensitive data between sender and receiver in real-time over a network"[6].

Real-time communication covers a wide variety of communication requirements depending upon the characteristics of the application involved. These communication requirements range from those needed by applications exchanging continuous data like audio/video isochronously to those exchanging signals in the control system applications [4]. These communication requirements are referred to as Quality of Service (QOS) requirements [4] [2][3]. Applications that involve real-time communication will be called real-time applications from now on. All real-time applications require a desired QOS from the network that supports them. This QOS can be measured in terms of upper bound on end-to-end communication delay, minimum amount of bandwidth, data throughput etc.

2.2 Real-time Applications

Based upon their QOS requirements real-time applications can be categorized as continuous-media or message-oriented applications.

2.2.1 Continuous-Media (CM) applications

These applications include the ones that generate data at regular intervals of time and, which need to be delivered in real time. The data might contain time-based information, like audio/video/animation, which should be played-back at a predetermined time-instant or else the information looses fidelity. Therefore, the network should preserve the playback or the display rate in order to conserve the significance of the information being delivered. The major QOS requirements for these kind of applications can be defined in terms of rate at which the data is transmitted over the network. A better and broader way to define QOS requirements of CM applications stated in [4] is that of "data stream abstraction, that is a simplex, end to end, continuous, sequenced, periodic transfer of data " [4] [2]. A more recognizable term for continuous-media applications is "Real-time streaming" applications. There are a number of popular applications generating real-time streams on the Internet. These one-way streaming applications can again be classified as either representing a pull model as in the case of one-to-one applications or a push model as in the case of multicasting applications. Applications like Internet telephony and audio/video conferencing in which the interaction is two-way represent two-way streaming applications.

2.2.2 Message-Oriented Applications

Message-oriented applications require block(s) of data to be delivered within a deadline. The data could contain information like control signals, text, commands etc. The information is time-based and is generated irregularly and in a limited amount. Unlike continuous-media applications, there is no timing relationship between the blocks of data being generated. These types of applications desire QOS in terms of meeting a prefixed deadline for delivering the data. An interesting example of this type of applications would be remotely controlling a robot.

2.3 Service Models

QOS is the quality of service provided by a network to the applications that it supports. QOS is determined by the service model that the network implements.

The service model constitutes the service commitments that the network makes to the data-flow. The service commitment describes the service provided by the network in response to a particular request. Based upon the way a service is characterized, service commitments can be divided into quantitative or qualitative commitment service [2] [1] [8].

2.3.1 Quantitative Service Commitment

A quantitative service commitment is an absolute assurance by the network that it will meet or better the agreed-upon quantitative specification(s). The commitments made by the network in this case are mostly in terms of minimum bandwidth that will be available to the applications. This type of commitment demands a certain type of admission control since bandwidth is always available in finite amounts [2][1].

2.3.2 Qualitative Service Commitment

A qualitative service commitment is an assurance by the network to provide a level of service to a particular flow, which is relative to the one provided to other traffic flows. This type of commitment is discriminatory. Higher-priority flows are given better quality of service as compared to the flows having lower priorities. This kind of commitment does not call for any admission-control mechanism unlike quantitative-service commitment. [2][1]

2.4 The Internet

Unfortunately, the Internet does not provide any kind of QOS guarantees to its applications. The data from applications is routed in packets to the destination through a number of routers. The Internet at present only provides best-effort service. The routers determine the best path for the data and this service is provided in first-in-first-out manner. However, there are no commitments or guarantees made by this best-effort service model in terms of QOS parameters such as delay, bandwidth etc [2][1].

The quality-of-service provided by this kind of uniform service to all, varies with the network traffic. The rate at which data is delivered decreases and the number of packets dropped increases directly with the rise in the network traffic. Obviously, this kind of service does not suit the real-time applications, which require a consistent level of quality from the network. However, the "best-effort" service model has served well in supporting applications which are tolerant of delay [2][1].

Recent advances in network technology have made the best-effort service model substantially reliable. This, in addition to the invention of new concepts in software technology like RMI, CORBA and data-streams have resulted in a surge of real-time applications like delivering voice/video over IP, remote X-terminals, data fusion etc. Tools like RMI and CORBA enable developers to distribute software objects over networks efficiently. The result of these technologies is that the Internet is increasingly being used for real-time communication, though with some compromises.

It is clear that a more sophisticated service model than the best-effort is required to truly serve real-time applications over the Internet. Any new model will have to provide some control over end-to-end delay introduced by the Internet [2][1].

2.5 Service Models for the Internet

There is a strong feeling among the Internet community that the current Internet infrastructure must be extended in order to support real-time applications. One obvious way to deal with this problem is to lay a new parallel network, which can support real-time services. However, this approach is not very feasible since it would be complicated and expensive to build and maintain such a network. Moreover, we will lose the advantage of statistical sharing between real and non-real-time traffic on a common infrastructure. Another way would be to extend the current Internet infrastructure to serve an integrated suite of services rather than just providing the "best-effort" service [1][2][3]. End-to-End delay is the central quantitative measure to determine the quality of service offered by the network.

IETF (Internet Engineering Task Force) set up a working group called "Integrated Service" to specify the enhanced service model and to define standards and necessary requirements to implement the new service model. This enhanced model, called the Integrated Services Model, defines five classes of service: best-effort, fair, controlled load, predictive or controlled delay and guaranteed service class. If supported by the Internet, these classes will be able to satisfy the requirements of a wide-variety of real-time applications.

2.5.1 Integrated Services Model

The five classes defined by this model are:

1. Best-Effort Service Class: As mentioned before, this class of service is already being provided by the Internet. The only guarantee that this class of service makes is that it will do its best to deliver data reliably and without adding any significant delay. This service-model serves the applications that are not delay-sensitive well.

2. Fair Service Class: This is a further enhancement to the best-effort service class. No requests for guarantees are made by the application but the network tries to partition the resources in a fair way.

3. Controlled-Load Service Class: This type of service class provides a guarantee to the user that it would appear to the user that there is a little traffic over the network. So the performance of the application being served by this class will approximate the performance with the best-effort service class under lightly-loaded conditions. In this type of class a limited amount of traffic is admitted. [19]

4. Predictive/Controlled Delay Service Class: This type of service class controls the delay that is experienced by a data flow. The application requests that its flow should not experience a delay greater than a specified maximum value. The network can than accept or reject this request.

5. Guaranteed Service Class: This service class provides a guaranteed upper bound on delay perceived by a particular data flow and an assured amount of bandwidth to the flow. [16]

All the service classes discussed in the previous section with the exception of best-effort, provide certain commitments to the conforming data flow. The best-effort traffic entering the router will not receive any such service commitment but will have to do with whatever resources are available. The level of service provided by these QOS classes to the flows will be programmable on per-flow basis according to the requests from the end applications. The requests by the end applications can be passed on to the routers by the network management procedures or by using the resource reservation protocols like RSVP (ReSerVation Protocol) [35].

Implementation of these service classes requires the implementation of the following set of methods. [1] [2] [8]:

1. Admission Control: An admission control mechanism is required to refuse service to applications if the requirements stated by them cannot be met by the network.
2. Resource Reservation: A resource reservation mechanism is required to pass the requirements specified by the end applications to the routers on the path to be traversed by the flow. RSVP [35], ST-I and ST-II are some of the protocols that implement resource reservation.
3. Policing: Policing is required to monitor the data flow in order to see that it does not exceed what has been allocated to it and/or what can be supported by the network.

2.6 Conclusion

The Internet might not be fully ready yet but it can still support real-time communication to some extent. Also there is lot of work being done on developing service models to enable the Internet to support real-time as well as other applications.

Chapter 3

COMPUTER-SUPPORTED COLLABORATIVE WORK

"*An identifiable research field focused on the role of computer in group work.*" [22]

"*An endeavor to understand the nature and characteristics of cooperative work with the objective of designing adequate computer-based technologies.*"[23]

"*the design of computer-based technologies with explicit concern for socially organized practices of their intended users.*" [24]

Introduction

As can be seen from the above definitions there is no one specific description of the term CSCW or Computer-Supported Collaborative Work. Since it was first coined by Irene Greif and Paul Cashman in 1984, CSCW has rapidly gained both popularity and notoriety because of its ambiguous and multi-faceted nature. The field of CSCW covers wide ranging fields from computer science to social and cultural studies and this has naturally brought together researchers from all these fields under one umbrella. This multi-disciplinary nature of CSCW is probably one of the reasons why there is no one agreed-upon definition.

3.2 CSCW – Different Views

There are four existing views about CSCW, which can be classified as follows [18]:

1. CSCW as a concept that brings together researchers from various fields to discuss ideas that are concerned with people, computers and cooperation of some sort.

2. CSCW as an opportunity for writing new kind of fancy computer systems that let people work in a group.

3. CSCW as a paradigm shift from the way the computer systems have traditionally been designed.

4. CSCW is a field that involves understanding of how people cooperate in real life and designing computer systems based on these understandings.

Howard in 1988 phrased the term "strict constructionist"[36], which refers to the researchers that have the perspective number 2 about CSCW. The resulting computer systems developed are referred to as groupware. Groupware is a short term for group software and it represents development of software that supports group work. Since the focus of this thesis's work is designing and development of real-time collaborative applications, we will take this "strict constructionist" view of CSCW.

3.3 Groupware

"Intentional group processes plus software to support them." Peter & Trudy Johnson-Lenz [25]

"A co-evolving human-tool system." Doug Englebert [25]

"Computer-mediated collaboration of person-to-person processes." David Coleman [25]

Interestingly, although groupware is considered as just one perspective of CSCW, the term groupware was coined much before the term CSCW came into existence. However, with the introduction of the field CSCW, groupware became just one aspect of CSCW.

Groupware is designed to support group work, in order to accomplish a particular task. In essence, Groupware supports the way in which a team or a group works.

Groupware computer systems can be classified into two categories based on the type of collaboration they provide across time. These two categories are:

1. Groupware systems that support synchronous collaboration. This type of groupware systems require that the user be present at the time of activity in order to participate. For example, in order to participate in a video conferencing session the participants have to be physically seated before the video camera.

2. Groupware that supports asynchronous collaboration. Unlike synchronous groupware systems, the user can participate in an activity any time he/she feels like. For example in order to participate in an e-mail-based discussion the participant can check his/her messages anytime he/she feels likes and respond.

Designing groupware systems is altogether a different ball game than designing single-user applications. We will get into some of the design issues later in this chapter but before that let's take a look at some of the reasons why we need Groupware.

3.3.1 Why Groupware?

The advantages of multi-user applications or groupware are enormous. Some of these advantages are listed below.

1. Groupware provides an infrastructure for interpersonal communication, which may result in a better, effective and quick solution to a problem.

2. Enables people to work from anywhere on the Internet/Intranet.

3. Groupware provides companies with a support base for interfacing with its geographically dispersed workforce.

4. Groupware is environment-friendly since it results in less travel by group members.

5. Owing to the fact that the intention of Groupware is to mimic human face-to-face communication, it allows people with non-computer backgrounds to participate in collaborative work easily.

6. It makes obsolete the need for all members to be present in the same room in order to collaborate on something.

3.3.2 Designing Groupware

Even though groupware is perceived as a technology-rich solution to CSCW, it still is very important to understand the social working of groups in order to design effective groupware applications.

Successful groupware applications are complicated to design because it takes more than just technology to do so. The ingredients to successful groupware are technology and social and cultural understanding of how people communicate.

3.3.3 Groupware Applications

This section goes through some of the popular Internet-based groupware systems.

3.3.3.1 E-mail Systems

The most basic and traditional of all groupware applications is the electronic mail. It facilitates asynchronous collaboration among team members. There are many e-mail systems available in the market today like cc:Mail, Eudora, Microsoft Exchange etc. The web-based e-mail systems like Hotmail have made it possible to check or send messages from anywhere in the cyber world. However, e-mail systems provide very limited collaboration. But, the e-mail system would still form an essential part of any groupware system. The success of e-mail systems depends on how efficient a user is in reflecting his/her thoughts in his/her messages.

3.3.3.2 Audio/Video Conferencing

Unlike e-mail systems audio/video conferencing supports collaboration in real-time. The team members are able to listen and/or even to see each other using these tools. The most important feature is that it all happens in real-time.

3.3.3.3 Desktop Sharing

Desktop sharing allows real-time data sharing among different machines. Two of the well-known desktop-sharing groupware systems are MS NetMeeting and PC Anywhere. Both of these applications let the user take control of a computer, which is connected to the network and whose IP address or name is known. These applications are useful for debugging systems remotely.

3.4 Conclusion

Groupware systems like e-mail, audio/video conferencing and remote desktop-sharing support real-time multi-user collaboration. However, none of the example groupware systems discussed supports every aspect of collaborative work. In addition, developing such a system from scratch would be a mammoth if not an impossible task. The field of networks and CSCW is still evolving and any system developed today might become obsolete in a short time as new discoveries and inventions are made. What we need is a framework, which not only supports quick and efficient development of multi-user tools but also hosts these tools over a common platform. This will let us write new applications and use them along with the already existing applications. In such a system, each application added will result in a better collaboration among team members. The ISAAC (Integration of Synchronous and Asynchronous Collaboration) framework, a project sponsored by Defense Advanced Research Projects Agency (DARPA) [37], intends to achieve exactly this.

Chapter 4

BUILDING REAL-TIME MULTI-USER APPLICATIONS

The Internet provides a huge potential for supporting real-time collaborative work (RTCW) applications. With the rapid growth of the Internet, both qualitatively and quantitatively, there is a strong future for such applications.

The choice of a tool for developing real-time collaborative applications would prove crucial for the success of any application. A collaborative application can be developed either from scratch or can be built-on a framework. In this chapter we will look at both of these methods and see how these two compare. During the course of this thesis work, we developed a collaborative text editor using the Habanero framework. Habanero is part of the ISAAC project, which handles the support for synchronous collaboration. Habanero provides developers with a common platform for both development and deployment of collaborative applications.

4.1 Essential Components of any RTCW application

Any real-time collaborative application should implement the following:

1. Support for synchronous collaboration.

2. Support for asynchronous collaboration where the collaboration is expected to take place over a long timeframe.

3. The basic communication infrastructure.

4. Implementation of the rules of collaboration e.g. floor control mechanism to force the participants to take turns.

5. A user interface that imposes minimum restriction on user activities.

6. Provision of hooks to extend the system.

7. Platform independence.

The task of developing real-time multi-user applications is far more complicated than their single-user counterparts. The development of these applications requires expertise in number of fields like networking, sociology, software engineering etc.

4.2 Building RTCW applications from scratch

This is how most of us, software developers, are used to programming. This method involves analysis of a problem, selection of the most appropriate tool, development, and finally the deployment of the application. In addition, this process is repeated for every problem.

4.2.1 Advantages

The major advantage of using this method is that it suits the traditional mindset of most of the software developers. This results in minimal costs in terms of training the developers.

Using this method the developer has complete control over the whole development process of an application and uses his/her discretion to decide what components to develop, how to develop them, and what would be the flow of control.

The flexibility enjoyed by the developer over the design and code of the application would most likely result in a code that is highly optimized and efficient. The level of optimization and efficiency will, however, depend on the amount of expertise and experience of the developer.

4.2.2 Disadvantages

Building multi-user applications from scratch also has some drawbacks.

Developing real-time collaborative applications from scratch means implementation of the essential components by the developer. The developer might be an expert in his/her own field but may not be very good at networking or understanding human-behavior. Moreover, developing all these components would prove to be a huge undertaking.

The collaborative applications developed using this method are not very accommodating to the future advances in the fields related to CSCW. Incorporating newly-discovered concepts into existing applications might result in rewriting the whole application from scratch.

Adding new features to an existing application would mean modifying the existing code, which means the developer should have extensive knowledge about the previously written code.

Every application would require implementation of the essential components. This result in reinventing and rediscovering the basics repeatedly, which eventually adds-up in terms of costs. The development-cycle for collaborative applications developed using this method is usually very long.

4.2.3 Development of Real-time Collaborative Applications

Developing complicated real-time collaborative applications from scratch is slow and tedious. The implementation of the networking component is a complex task in itself. The other tasks like distribution of information and providing some kind of an arbitrator to ensure every participant follows some rules are again not very easy.

However, recent advances in Java technology like object serialization, networking and RMI (Remote Method Invocation)) have made some of the tasks involved in building real-time collaborative applications easier. Java has extensive support for networking where the developer does not have to worry about the underlying network architecture in order to write network-based applications. This makes the implementation of the communication component of any real-time collaborative application relatively easier. Java's object serialization and RMI technologies make it easier respectively, to add persistency into objects and distribute them over a network respectively.

Although Java provides solutions to augment the drawbacks presented by development of collaborative applications from scratch, this method is still only suitable for developing small collaborative applications.

Moreover, building collaborative applications from scratch is fine if we are developing an isolated application, which cannot be classified into any particular domain of applications. The applications designed for a common environment or domain, e.g. automotive and telecommunication applications, tend to share some common core concepts and architecture. And, most of the cost and effort involved in developing these applications is a result of redesigning, reinventing and re-implementation of these common concepts and design. And, in case of complex applications which work across heterogeneous hardware architecture, operating systems and platforms, it becomes extremely "difficult to build correct, portable, efficient, and inexpensive applications from scratch [26]". Hence it becomes imperative to look for alternative methods that offer features like code and design reuse.

The class libraries, which provide a set of already-implemented class objects, offer an obvious alternative. The developers instantiate ready-made class objects and extend their functionality by using either the process of inheritance [33] or delegation [33] to achieve the objective. Although class libraries provide a way to reuse code, it still requires the developer to call individual class objects and connect them together to perform a task. Thus the use of class libraries does not provide a way to reuse designs and ideas.

With the use of class libraries, the developer is still responsible for the flow of control that occurs in the application. The handling of flow of control becomes complicated when dealing with collaborative applications. This is so because this requires implementation of objects like observers and arbitrators, which decide the events and the order of events that is shared among the participants of a session.

The frameworks provide us with another alternative for developing real-time collaborative applications without repeatedly reinventing the wheel.

4.3 Building RTCW applications using Frameworks

4.3.1 Frameworks

"A Framework helps developers provide solutions for problem domains and better maintain those solutions. It provides a well-designed and thought out infrastructure so that when new pieces are created, they can be substituted with minimal impact on the other pieces in the framework."[28]

Frameworks target a particular domain of applications. Frameworks are implemented using a particular programming language and hence inherit the characteristics of that language e.g. object-oriented(OO) frameworks support concepts like code reuse, encapsulation etc. OO frameworks are more popular because of the benefits that any OO programming language carries and from hereon everything discussed about frameworks will be in the context of OO frameworks.

Frameworks provide a set of components that are generic to the domain of applications that is being targeted by the framework. The applications developed using this framework are able to interconnect these components to perform a particular task. Frameworks also provide an environment to develop new components to extend their functionality. In fact frameworks can be classified as being white-box or black-box according to the techniques used to extend them [26]. White-box frameworks are more open whereby the developers are required to know the inner implementation details. These frameworks are usually extended by using OO concepts of inheritance and dynamic binding on the hook methods provided by the frameworks. Black-box frameworks on the other hand encapsulate their implementation details and provide a set of interfaces to be used by the developers to plug-in their components [26].

In essence, a framework is a base-application that can be used by developers to build their applications upon. Frameworks provide an infrastructure that is designed to support the development of a particular class of applications e.g. a framework for developing automotive applications or a framework that supports telecommunication-related applications. Java's RMI (Remote Method Invocation) and the ORB (Object Resource Broker) frameworks support applications that require distribution of objects over a network. Java's RMI provides an interface that is used by objects to call other objects over a network. This interface hides the details of the complexities of the underlying network communications mechanism from the developer. This is where the real strength of frameworks lies, encapsulating the generic components and providing a simple but stable interface/hook methods to these components [26].

Frameworks are semi-implemented applications whose functionality can be extended/specialized to create new applications [26][38][39].

Frameworks might be viewed as class libraries but at a lower level of abstraction [27]. This means frameworks provide objects that are more concrete unlike in the case of class libraries. Frameworks can afford this lower level of abstraction since they are designed to support a particular type of applications and hence they do not need to be as general as class libraries, which try to accommodate wide variety of application types.

Frameworks unlike class libraries not only contain class objects but also the connections between these objects in order to perform a particular function.

Frameworks, unlike class libraries provide developers with an infrastructure to build a particular type of application. By infrastructure, we mean actual implementation of features or components that are common to the class of applications being targeted by the framework.

4.3.2 Frameworks and Design/Code Reuse

"Frameworks are an object-oriented reuse technique."[27]

The reuse can be in terms of both design and code. As already mentioned before, frameworks provide a set of generic components. The use of these components by other applications represents code-reuse whereas inter-connection of these components to perform a particular function encourages design-reuse [28].

The use of class libraries unlike frameworks only encourages code-reuse and not design reuse. A framework not only provides implemented class objects to be reused but also acts as a template that can be used during the design phase of the application.

The concept of code-reuse reduces the development time of an application considerably. Any well-implemented framework is designed using well-tested techniques like design patterns [33]. Therefore applications developed within the frameworks inherit a good design along with the framework components for code reuse.

The design reuse results in the development of components that are similar to each other since they use the same framework as a template. This uniformity [27] results in applications with similar structure and hence makes the applications easier to debug for the developers. And in case of a GUI framework it will help to generate user interfaces with similar look and feel.

Frameworks also encourage the reuse of domain-specific expertise [40]. The domain-specific generic components provided by a framework are designed by the developers who are experts in that particular field.

4.3.3 RTCW Frameworks

RTCW frameworks are the frameworks that facilitate the development of RTCW applications. This type of framework provides us with an actual or partial implementation of the components that are generic to any RTCW application. Such a framework enables a developer to focus more on the actual application than to worry about the lower-level intricacies of collaboration.

4.3.4 Benefits of using Frameworks

The use of frameworks provides the developers with the following benefits [26]:

1. Modularity: The frameworks provide a set of simple and stable interfaces to its components, which are used by other components or objects to call the services on these components. This ensures encapsulation of objects whereby

all the implementation details of the object are hidden by the interfaces. This in turn results in modular objects that are easy to maintain [26].

2. Reusability: As discussed before, frameworks provide a set of implemented components that are generic to the class of application that is being targeted by a framework. These components are designed and coded only once and use some time-tested techniques. This results in code-reuse and it drastically reduces the implementation time of an application [26].

3. Extensibility: The frameworks provide hooks to its components, which can be used by developers to extend the functionality of these objects. Therefore, the infrastructure provided by a framework is highly flexible and can be extended if required by an application [26].

4. Inversion of Control: The framework itself executes the flow of control in applications developed. This is a powerful feature of frameworks especially in case of RTCW frameworks where correct flow of control is crucial and complicated in order to make objects and actions shareable.

4.3.5 Drawbacks of using Frameworks

Frameworks enable us to develop applications that are modular, extensible and are compatible with each other. However, there are a few drawbacks of using frameworks some of which are listed below:

1. Developing a framework is very challenging and takes a lot of coding before you can even start writing an application [26]. Hence, the benefits of using frameworks are long term. This is not a problem with this thesis work because an existing framework was used.

2. The developers need to change their mindset in order to use frameworks because developing applications using frameworks is very different from

developing them from scratch. The major difference that should be noted is that frameworks control the flow of the application, which is totally different from the traditional style of programming where the application controls its own flow. The framework, especially one that supports building of collaborative applications, has objects like observers and arbitrators that control the flow based on the events generated by the participating clients.

3. The developers need to familiarize themselves with the basic infrastructure and hooks that a framework provides for them to start using it efficiently and effectively.

4.4 Conclusion

Building applications ground-up gives more flexibility to the developer but, at the same time, it does not encourage good programming practices. Frameworks, on the other hand, provide us with an alternative where methods and properties inherent to a particular domain of applications are already implemented. Using frameworks can cut the time required to develop applications significantly.

Chapter 5

A FRAMEWORK FOR INTEGRATED SYNCHRONOUS AND ASYNCHRONOUS COLLABORATION (ISAAC)

5.1 Introduction

ISAAC is a DARPA-funded project, whose main objective is to make "computer-assisted collaboration in science, engineering, and real-time decision support more natural, more powerful, and more responsive to the multi-modal communication needs of users" [15]. ISAAC also intends to make collaboration possible across platforms and time. The collaboration across time enables people to participate in a session synchronously or asynchronously. Moreover, platform independence enables participants operating on different platforms to collaborate with each other.

This system enables participants to work in a persistent-information immersion environment. This information can be used and modified by a participant. The information would always be persistent even if the original source is a temporary one.

5.2 ISAAC Components

ISAAC is still work in progress. However, when finished it will consist of [15]:

1. Support for synchronous or real-time collaboration.

2. Support for asynchronous collaboration.

3. Persistent Object Store (POS) component.

4. Event notification and event/action services.

5.2.1 Support for Synchronous Collaboration

The synchronous collaboration requires each participant to be virtually present in order to participate in a session. This component of ISAAC is built on Habanero and will be covered in detail in the next chapter.

5.2.2 Support for Asynchronous Collaboration

The Habanero component will be extended to enable participants to participate in a session at a time that is convenient to them. Already, Habanero enables participants to record a session and play it back at a later time. However, the support for asynchronous collaboration would be extended further with mechanisms like forwarding selected events to participant's e-mail or maybe something more advanced like an agent that responds back on behalf of the participant [15].

The participant will be able to leave/join a session at his/her will and still not loose any information or events. This makes collaboration very natural and very close to the way human beings collaborate in real life. This also gives the participants the flexibility by not putting any restrictions on their activities.

5.2.3 Persistent Object Storage

The Persistent Object Storage or POS is required to store objects generated or used by a Habanero session. This is required for the persistency of information in order to support collaboration across time. The POS in ISAAC would provide [15]:

1. A place for storing objects mainly generated during a session. However, this would also be used to store external objects like Lotus notes, database objects

etc. This will enable participants to integrate their existing systems as a back-end to a Habanero session.

2. A place to store object-metadata. The metadata stores information about the objects, which can be used for retrieving these objects from a repository.

3. A facility to search the POS.

The POS could also have an API (Application Programming Interface) to perform the following functions [15]:

5. Connecting to a POS.

6. Add/Delete/Change/Get objects.

7. Getting object metadata.

8. Searching for objects.

9. Notification mechanism to notify of object changes to the concerned participants.

Adding POS to the current version of Habanero would extend its functionality to provide the following additional features [15]:

1. A place to store sessions.

2. The object-metadata information can be used to replay sessions.

3. A facility to search session-archives.

4. Indexing and sorting objects returned from a search.

5. Methods to create POS objects as well as use the existing ones

The following figure depicts how POS would fit in ISAAC along with Habanero.

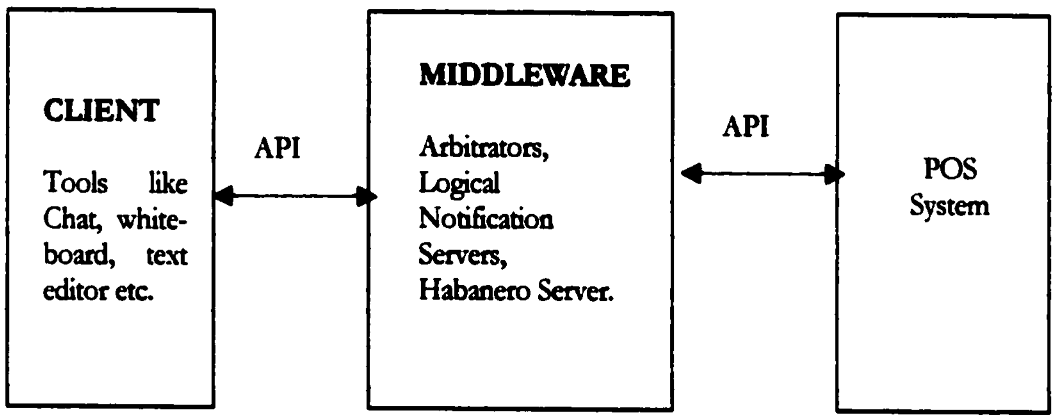
CLIENT
Tools like Chat, white-board, text editor etc.
API
MIDDLEWARE
Arbitrators, Logical Notification Servers, Habanero Server.
API
POS System

Figure 5.1

The POS at the back-end would let us include powerful information services in the sessions. The POS would be accessed like a black box where a participant would not need to know the types of POS's available to the session. The access to a POS would be provided through a common API. This common interface to all the POS would enable access to information, which is independent of the type of POS being accessed.

5.2.4 Event Notification

The current Habanero event notification is restricted within the session in which it was generated. However, it is planned to extend event notification across sessions [15].

The new notification mechanism would have the following features [15]:

1. Timer-based action events. This feature can be used to automatically generate an event at a particular time.

2. Broadcast of events to other sessions. This feature will enable collaboration that goes across multiple-sessions.

3. Notification of change in a POS. This event will be generated when something of interest to a participant or a session changes.

5.3 Current Work in Progress

ISAAC is in its final stages for its first public release sometime before the year 2000. The work is currently to develop or acquire components to provide integration of synchronous and asynchronous collaborations.

5.4 Conclusion

The ISAAC project intends to bring together asynchronous and synchronous collaboration in an information-immersion environment. It would support seamless collaboration across time, space and platform.

Chapter 6

HABANERO

6.1 Introduction

Habanero is a framework that supports the development of real-time collaborative applications for the Internet. Habanero was initially designed to facilitate collaboration in the fields of science and education [13]. However, it has the potential to be used in numerous other fields.

The Habanero framework is implemented using Java and it depends on sharing of Java objects in order to support collaboration. It is claimed by the Habanero team that the applications developed using this framework are scalable to any number of users. Moreover, since Habanero is implemented using Java, it is an OO framework which is platform independent.

Habanero uses a client/server architecture and the concept of replication to distribute information. All the information to be shared is represented in the form of Java objects. Each collaborative session taking place within Habanero has one server running and all the participants are connected to this server through individual clients. An object received by the server from any client is replicated and sent to each of the connected clients.

Habanero can be divided into two parts – the Habanero framework and the Habanero environment. The Habanero framework supports the development of real-time multi-user applications whereas the environment is responsible for hosting these applications [9].

6.2 The Habanero Environment

The Habanero environment consists of a server, clients and a set of collaborative tools. The client and server act as an infrastructure for hosting collaborative sessions. The collaboration is accomplished by replicating the desired states of an application among all of the clients.

6.2.1 The Server

The server consists of serializers, arbitrators and routers. As already mentioned before, the information in the Habanero framework is represented in the form of Java objects. These objects are shared among the clients through a server in response to an event that is designated as "to be shared". When a new client joins an ongoing session the server synchronizes it with the current state of the session. This synchronization is done by replicating the session state on the client [13].

The events are generated in response to a client's action. The actions are considered to be the medium of exchange in Habanero [13]. When a "to be shared" event occurs, it must be propagated to all the desired clients. The sequence in which events occur must be respected in order to have any kind of meaningful collaboration. This sequence is maintained by the server with the use of a component called an arbitrator. An arbitrator, implements the rules of a multi-user application in terms of what and when anything is executed. The server assigns an arbitrator to each application. This arbitrator can be extended using the Habanero API to do all sorts of things like implementing rules specific to a particular application. For example, in case the clients are connected at different bandwidths than an arbitrator can set a maximum speed that is suitable to the client connected at the lowest speed. A customized arbitrator can be used to develop a multi-user game like checkerboard where the arbitrator forces rules of this particular game on clients. The arbitrator can be considered as the central point of control, which ensures each client plays by the rules. By default, the arbitrator is responsible to see that events shared are in sync with the original order of events [9].

Just simple sharing of events is not very useful without the ability to share information among clients. Habanero uses serializers to achieve this sharing of information or objects. Habanero uses Java's object-serialization feature to implement its serializers. The serializer serializes and de-serializes objects to write objects to the output stream and read objects from the input stream respectively [9].

The router is responsible for channeling all the shared events and objects to the desired clients. The router or the communication manager is implemented in the Habanero server but it can also be programmed to work on the client side. This gives the developers a flexibility to develop applications with their own communication managers [9].

The arbitrator, serializer and the router are the most important components that are provided by Habanero. In addition to these components, Habanero also provides support for networking and security. Habanero only supports TCP/IP communication at present, but this will probably be complemented with the support for multicasting. The only level of security that Habanero provides currently is that it does not let unauthorized clients to connect to a session [9].

6.2.2 The Client

The client interacts with a single server, which in turn communicates with all other clients. The server communicates with an application on the client through the application's proxy. There is a proxy for each application on every active client. Moreover, all the proxies for the same application have identical names. This allows the server to communicate with the correct application for sharing of events and objects. It is interesting to note that clients also implement serializers, arbitrators and routers. These components act as respective proxies for individual applications and forward the events to their counterparts on the server [9].

The Habanero client provides a graphical interface to the Habanero environment. The client lets the user select a session of his/her choice and connect to it. And once connected the client synchronizes itself with the current state of the session.

The client provides the following features:

1. Lets the user create/edit a certificate for identifying himself/herself to other connected users.

2. Provides the user with the powerful feature of capturing an ongoing session for replaying later. Either the recorded session can be played locally or it could be distributed. These sessions can be stored using the POS technique discussed in the last chapter and can be played from there at any time.

3. Allows the user to notify other participants to join the session. This notification can made either synchronously i.e. through the client itself or asynchronously i.e. through the e-mail.

4. Lets the user identify and locate the participants of the session.

5. Lets the user create his/her own session or join an existing session.

6. The client lets the user terminate a session in number of ways. The user has option to do the following:

    a) Terminate its own state without affecting the state of other participants.
    b) Kill the session totally.
    c) Leave the applications on but deactivate the session.

The client also provides the user with important session information like the list of participants, network settings etc.

6.2.2 The Tools

Habanero comes with a set of pre-implemented tools like a whiteboard, shared browser, mpEdit editor, chat etc. The user can develop his/her own tools and integrate them within the Habanero environment.

6.3 Habanero Framework

The Habanero framework consists of an API, which is used to write multi-user real-time applications that run within the Habanero environment. Additionally, this API also lets developers convert their existing single-user applications into multi-user applications in a short time.

In addition to this API, the framework also includes pre-implemented arbitrators, serializers and network components. Therefore, the developer does not have to learn or develop these components and he/she can focus on the actual application. The developer only needs to implement four methods in order to plug-in his/her applications into the Habanero environment. These methods are responsible for:

1. Defining how the application is displayed when instantiated.

2. Processing events generated by clients.

3. Reading the current state of an application from the input stream.

4. Writing the current state of the application to the output stream. This is called marshalling.

The Habanero API also provides classes and methods to extend the functionality of the components used by the Habanero environment like arbitrators and routers.

The Habanero framework also includes a porting wizard that can be used to rapidly convert single-user Java applications to multi-user applications. The wizard is smart enough to extract components and events from a Java program and make these components shared if asked to do so.

Habanero lets the developer decide what objects and events in an application are to be shared. The objects that are designed to be shared are required to implement a particular interface. In addition to sharing of events, Habanero also lets the developer write his/her own events and make them sharable. This is a powerful feature for applications like a text editor where sharing every key press event could easily overload the system. Instead of sharing every key press, it would be more efficient to share a block of text. In order to do so, the developer can write an event that sends the block of text to the server. This event than could be associated with a button press or a menu select action.

6.4 Habanero API

Habanero provides an extensive set of classes. These classes are divided into the following packages.

1. Set of classes to connect to MBONE (Multicast Backbone).

2. Extended set of Java's standard GUI component classes.

3. Set of classes to manipulate the environment in which a session takes place.

4. Set of classes that are used to develop collaborative applications.

5. Set of classes that support the use of different protocols like RTP (Real Time Protocol). This package is not developed yet.

6.4.1 MBONE

MBONE is a virtual network that lies on top of the Internet and serves as an experimental test-bed for applications using multicast IP (Internet Protocol). The multicast IP unlike the traditional unicast IP enables the data to be addressed and transmitted to multiple recipients at the same time. This leads to substantial savings in terms of consumption of bandwidth by Internet-based multimedia applications.

Habanero provides functionality to advertise a session to the MBONE community and to connect to this multicast Internet backbone. Unfortunately like most of the other packages in the Habanero API, not much information is available to fully understand this set of classes.

6.4.2 GUI Components (AWT package)

This package provides a set of GUI components, most of which are extended versions of Java's standard GUI components. Some of the components included in this package are a color palette, extended version of Java's standard buttons with the ability to contain images, windows to display messages and a enhanced list component.

6.4.3 Environment Manipulation (ENV package)

This package contains a set of classes that can be used to extract and manipulate the information about the environment under which a Habanero session takes place. The functionality provided by this package includes the ability to retrieve the profile and other information about the participants and to add/remove the tools from a session. The classes in this package can be used for the purposes of making the participants aware of who they are collaborating with and how to contact them.

6.4.4 Collaboration API (habanero package)

This package is the core of the Habanero API. This set of classes provides the functionality to develop collaborative applications for the Habanero environment and also to convert existing single-user applications into multi-user ones. Most of the classes used during the development of the text editor developed during the course of this thesis work are from this package.

6.5 The Habanero Porting Wizard

The Habanero porting wizard is designed to transform existing single-user Java applets into collaborative applications that can be run within the Habanero framework. The wizard takes the Java code as an input, recognizes all the objects and events in that code and makes these entities shareable if desired by the developer. The wizard is not powerful enough yet to work on complex applets. However, it acts as a good starting point for developing collaborative applications.

6.6 Conclusion

Habanero is a powerful tool that not only supports development of collaborative tools but also their deployment. The Habanero framework provides us with a powerful API to write RTCW applications for the Internet. Habanero is definitely a step in the right direction towards achieving a valuable framework for constructing real-time collaboration application.

Chapter 7

THE MULTI-USER TEXT EDITOR-IMPLEMENTATION

Almost every profession requires the creation of text documents. The complexity of these documents has increased significantly in recent years due to the rapid redefining of how organizations work. Many documents are no longer authored by a single person but are prepared by contributions from different areas of an organization. This requires collaboration among a number of concerned individuals to come up with the final document. Traditionally this collaboration has been done by distributing the tasks associated with a particular job among the individuals. This is a tedious way of collaborating and is prone to conflicts that might arise from the lack of appropriate communication among the contributors. Moreover, the globalization of organizations has resulted in a geographically distributed workforce and this has resulted in demands for more sophisticated collaboration [17].

There are some tools, like MACE, GROVE, SASE, SASSE and Lotus Notes, available that support collaboration for creating documents. As part of this thesis work, a real-time collaborative text editor was developed using the Habanero framework. The main objective behind developing this tool was to assess the abilities of Habanero to develop real-time multi-user applications and not to create a better editor.

7.1 Design Requirements

We used the extensive set of requirements stated by Michael Koch in [17]. This was done because we believe an external set of requirements was necessary for an unbiased analysis of Habanero.

The requirements are as follows:

1. The text-editor should be able support collaboration in a multi-user environment. However, the text-editor should also be usable by a single-user without any restrictions.

2. The text editor should have a feature, which would let the users work on his/her piece before sharing it with others.

3. The text editor should be able to store the history of updates to a particular document.

4. A participant should be aware of the presence of other participants.

5. The text editor should provide some sort of communication among the participants.

6. The text editor should be flexible enough to be integrated with external tools.

7.2 Architecture

The architecture of our text editor involved design and implementation of the following components:

1. **Arbitrator:** Since our editor will be used by multiple users simultaneously we would require to implement some rules in order to ensure constructive and meaningful collaboration. In our editor we will only allow one person to edit

the document at a time. And to achieve this we need a mechanism that will enforce the rules of collaboration and we do this by using an arbitrator that is an extension of Limit Arbitrator that comes with Habanero. By default in the Habanero framework when an event that is declared as shareable occurs it is replicated and propagated to all the clients by the server. The Limit Arbitrator can be used by a participant to lock these events from being shared if they are generated by other participants. In our editor we use this facility of locking to make sure only one person edits a text document at a time.

2. **Lock:** The arbitrator uses this lock to make sure only one participant has the control to the shared area of the text editor. The key to this lock acts as a token, which can be grabbed by any participant as long as no one else owns it. All the participants are made aware of the identity of the current owner of this key. The participants can then use one of the many Habanero tools like a chat line to contact the owner to ask him/her to release the key.

3. **Customized Event.** In order to collaborate over a text document we need to generate an event in response to any one of the following actions.

    a) A new file was opened.

    b) A block of text was selected.

    c) The text was modified.

   We designed a custom event that is generated in response to any of the action

   stated above. This event encompasses the type of the event and the

   information to be shared in response to this event. The event when generated

   is sent to the server which replicates this event object and distributes it among all the clients.

4. **GUI components:** The GUI for our editor contains a shared text component, button controls for obtaining/releasing the key, a scratch pad, a list component displaying the names of all the participants and a label displaying the current owner of the key to the lock held by the arbitrator. The shared text component is the text area, which contains the document that is being built collaboratively. Any modification done to the text contained by this component is seen by every participant. The scratch pad is the text component whose content is local to the participant. This area can be used by the participants to work on a piece of text before they add it to the final document. The list displaying the names of all the participants is an important component, which helps developers to know about their team members. The label showing the name of the current owner of the key helps other participants to identify the person editing the document.

The following block diagram represents the architecture used to develop the text-editor.

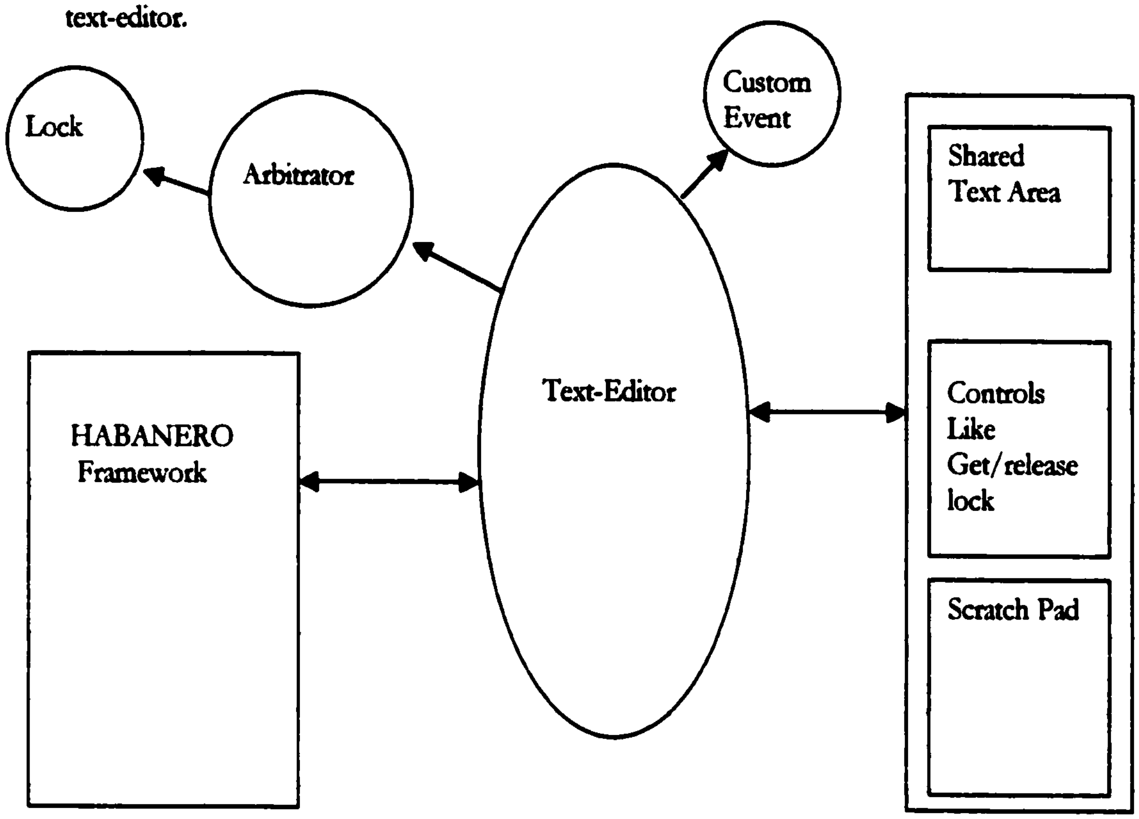

Figure 7.1

7.2 Implementation

The components discussed in the last section were developed using APIs provided by Java and Habanero.

The shared text area and the scratch pad use the TextArea object of Java's AWT API. The drawback of using TextArea object is its limited formatting ability.

We created a package called 'CollabTextEditor' under which all our .java files are located. In the directory where we installed Habanero there is a subdirectory called **apps**. Under **apps** we create another subdirectory named

'CollabTextEditor' so that Habanero will be able to find all our code files when we run our text editor within the Habanero framework.

Our main .java file is the TextEditorFrame which contains information that the Habanero framework needs for initialization when our 'hablet' is run from the framework. The term hablet means an applet that runs from the Habanero framework.

public class TextEditorFrame extends Hablet

In this class we override Hablet's startInFrame method mainly to declare the window parameters like its size and title, and calls to the hablet's init method, as follows:

```
public void startInFrame(MirrorFrame f)
{
    .
    .
    .
    f.setTitle("Text Editor");
    f.add("Center", this);
    f.setSize(700,500);
    f.addEventCode( Event.ACTION_EVENT );
    f.setLayout(new BorderLayout());
    f.show();
    this.init();
    .
    .
}
```

Now when the client joins a session for the first time, the server copies the current state of our text editor and passes it to the client. This is required to get the client in sync with the current state of an application. This is done by implementing the methods readHablet and writeHablet.

The readHablet is used to get the current state of an application when the client first starts. This method in our editor hablet reads the TextEditor object from the input stream. This object contains the current state of all the GUI components of our editor. The readHablet than extracts the current content of the text area component and copies it to its local counterpart.

```
   protected void readHablet(MarshallInputStream in) throws IOException,
                                                    ClassNotFoundException
   {
// Read the Text Editor object
      TextEditorObject = (TextEditor) in.readObject();
// Get the current content of the text component.
      txt = (String) (TextEditorObject.sharedArea).getText();
// Fill the local text component.
      (TextEditorObject.controlArea).releaseLock.setEnabled(false );
      inited = true;
   }
```

The writeHablet method is used to write the current state of our editor to the output stream.

```
   protected void writeHablet(MarshallOutputStream out) throws IOException
   {
      out.writeObject(TextEditorObject);
   }
```

Another key method of our main class is the processEvent(AWTEvent e) method. This method processes all the events received by our editor from the server. For our editor we implemented a CustomEvent event class and this is the only event processed by our processEvent method. This method processes an event only if it is not generated by the participant. This is done because the server sends an event to all the clients including the client that generated that event.

```
public void processEvent( AWTEvent e )
   {
```

```
//Check if the event was of type CustomEvent

     if( e instanceof CustomEvent)
     {
//Check if the event was generated by the local copy of the editor.
        if(! sentByMe )
        {
          CustomEvent he = (CustomEvent) e;
//New file was opened.
          if( he.getUniqueID() == OPEN_FILE_ACTION )
          {
           (TextEditorObject.sharedArea).setText( (String) he.getArg() );
          }
//A block of text was selected
          else if( he.getUniqueID() == SELECT_ACTION )
          {
           ObjectWrapper o = (ObjectWrapper) he.getArg();
           (TextEditorObject.sharedArea).select( o.getStartPos(), o.getEndPos() );

          }
//A request is made to the arbitrator to obtain the key to the lock
          else if( he.getUniqueID() == GET_KEY_ACTION )
          {
           System.out.println("acting on GET KEy action.");
           (TextEditorObject.generalArea).isKeyAvailable = false;
           ((TextEditorObject.sharedArea).txtArea).setEditable(false );

           String name = (String) he.getArg();
           ((TextEditorObject.generalArea).lockInfo).setText( name + " has the lock.");
            ((TextEditorObject.controlArea).transfer).setEnabled(false );
          }
//Key was released by the client.
          else if( he.getUniqueID() == RELEASE_KEY_ACTION )
          {
           System.out.println("acting on Release KEy action.");
           (TextEditorObject.generalArea).isKeyAvailable = false;
           ((TextEditorObject.sharedArea).txtArea).setEditable(false );
            ((TextEditorObject.controlArea).transfer).setEnabled( true );
          }
//Text was modified in the shared text area component.
          else if( he.getUniqueID() == KEY_TYPED_ACTION )
          {
            System.out.println("acting on Release KEy action.");
            KeyWrapper o = (KeyWrapper) he.getArg();
            ((TextEditorObject.sharedArea).txtArea).setEditable( true );
            ((TextEditorObject.sharedArea).txtArea).setText( o.getKey() );
            if( !TextEditor.hasKey )
              ((TextEditorObject.sharedArea).txtArea).setEditable( false );

          }


        }
        else
        {
          sentByMe = false;
        }
      }

   }
```

Any event that is supposed to be shared among clients should be explicitly declared as shareable during the initialization of the hablet. As shown in the code snippet below we only declare the CustomEvent class as shareable for our editor.

```
  public TextEditorFrame()
  {
     .
     .
     enableEvents(CustomEvent.HAB_EVENT_MASK);
  .
  .
  }
```

The CustomEvent class object is used to distribute data through the Habanero server when the user performs a particular action. This event object identifies itself to the event-handling routine with a unique identification number. This event object also contains an object variable, which stores the data to be shared among the participants. The CustomEvent's constructor takes the source, an id and an object as input parameters. The CustomEvent instances will have unique 'id' for different actions. The 'Object arg' parameter contains an object that contains information that is to be shared in response to a particular event instance. The methods getArg and getUniqueID are used to retrieve the corresponding entities.

```
public class CustomEvent extends AWTEvent implements StorageMode
{
  .
  .
  .
  public CustomEvent(Object source, int id, Object arg, int uniqueID)
  {
    super(source, id);
    this.arg = arg;
  }

  public Object getArg()
  {
    return arg;
  }

  public int getUniqueID()
  {
    return uniqueID;
  }
  .
  .
  .

}
```

When a user selects a block of text and he/she holds the key to the arbitrator lock then this select action should be replicated on all clients. So a select action generates a CustomEvent instance with a unique identification number and an object that contains the start and end position of the selected text block. We designed a ObjectWrapper class shown below that generates an object that contains the start and end positions.

```
public class ObjectWrapper implements Serializable
{
   .
   .
   .

   public ObjectWrapper( int start_pos, int end_pos, int action )
   {
      this.start_pos = start_pos;
      this.end_pos = end_pos;
      this.action = action;
   }
   public int getStartPos()
   {
      return start_pos;
   }
   public int getEndPos()
   {
      return end_pos;
   }
}
```

7.3 Working of the Text Editor

The block diagram below demonstrates the working of the text editor during an active collaborative session.

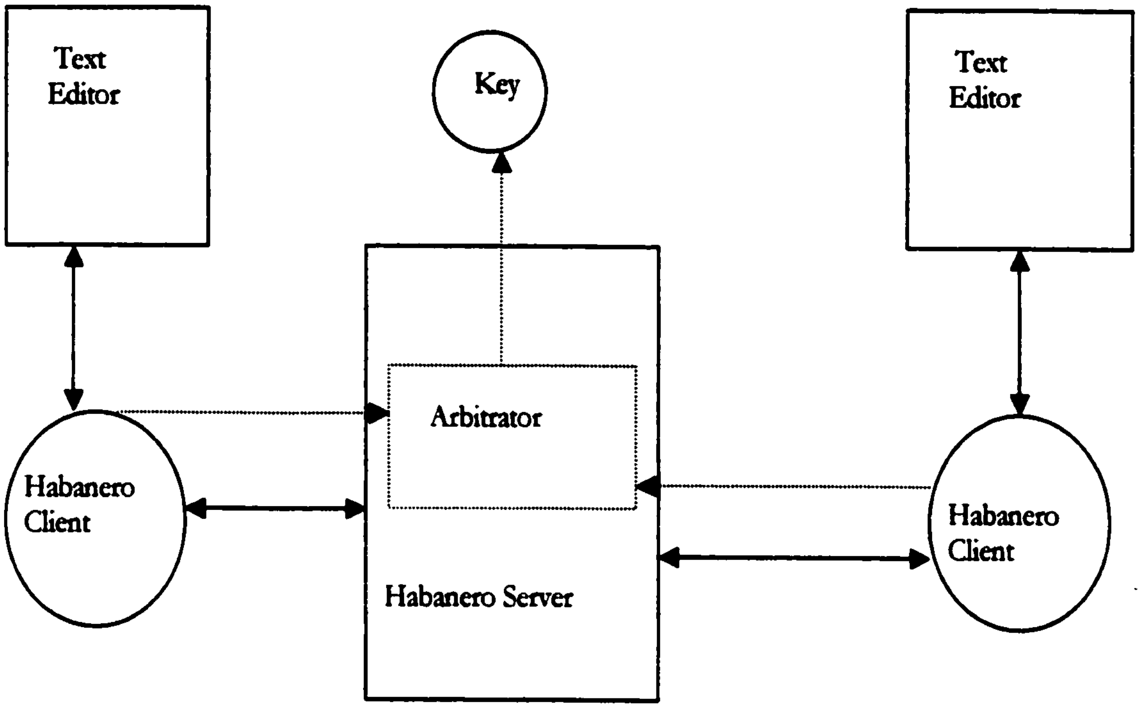

Figure 7.2

The session begins when a user instantiates the Habanero server. The server's IP address or the name is used by other participants to join this session. When a participant joins a session, he/she sees a window similar to the one shown in the figure below.

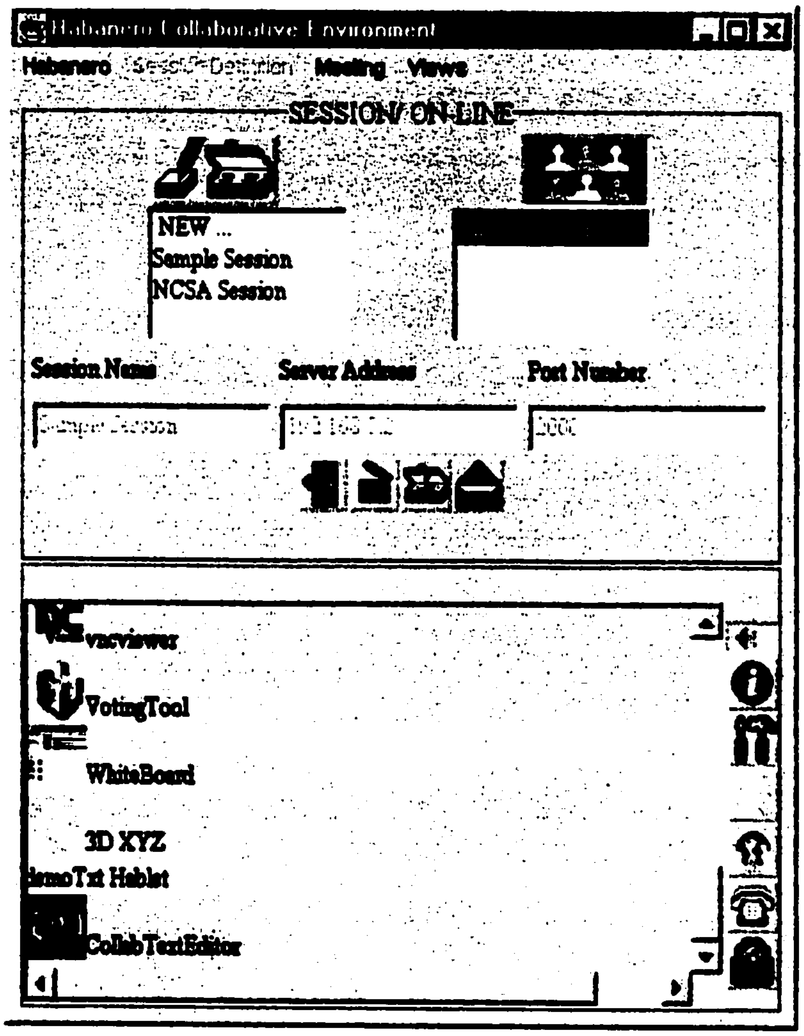

Figure 7.3

Any participant can than start a text editor session by clicking on CollabTextEditor" icon in the tools list. This will instantiate the editor for all the clients on their respective machines. One of the participants can than grab the key to start typing into the shared area. The key can be obtained by clicking on the "Get Lock" button. He/She can later release the key for others by clicking on another button labeled "Release Lock".

7.5 Screen Shots of the text editor

Below is the screen shot of the text editor in action.

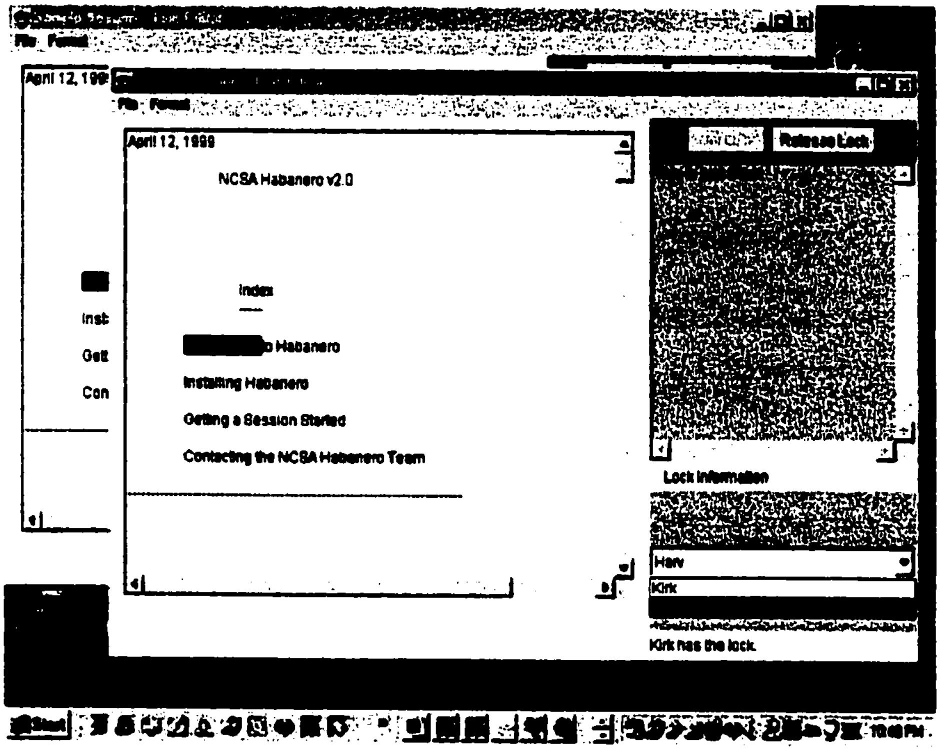

Figure 7.4

The drop-down box shows that currently there are two participants in the current session: Kirk and Harv. A label in the bottom right shows that Kirk currently owns the key. Kirk can then press the button "Release Lock" to release the key to make it available for another participant to obtain it. The scratch pad is on the right side below the "Release Lock" and "Get Lock" buttons. The shared area is the one on the left side with a white background. The scratch pad is next to the shared area on its right side.

7.6 Conclusion

We were able to fulfill almost all the design requirements except for creating a history for all the updates that happen to the text document. But we believe this can be easily achieved when the Habanero is enhanced with a POS back-end system. Moreover, the users can always use Habanero's replay facility to view all the updates. The text editor also does not contain powerful formatting features like most of the commercial text editors available today. But this we believe will not be a problem once Habanero fully evolves over a period of time and becomes stable enough to support powerful Swing or any other third party GUI components.

Chapter 8

STATEMENT OF FACTS/OBSERVATIONS MADE DURING THE DEVELOPMENT OF THE TEXT EDITOR

1. After we familiarized ourselves with the Habanero framework, we were able to develop our collaborative text editor in about ten days. We believe it would have taken us at least four times longer if we had developed the editor from scratch. Moreover, significant amount of time would have been spent designing and developing non-text-editor related functionality like networking and distribution of objects.

2. We believe using the Habanero framework resulted in a collaborative text editor that is far more efficient and stable than it would have been in case it was developed from scratch. This is because the framework provided us with a solid infrastructure for performing collaboration.

3. Our text editor is able to work along with any other tool developed using the Habanero framework. This eliminated the need for developing a secondary tool like a chat application for our editor since some other developer had already developed it.

4. Initially we developed a prototype for a single-user text-editor. The intention was to use the porting wizard to convert it into a multi-user text-editor. But, the wizard failed to recognize the text components in the text editor. However, the wizard restructured the code of our prototype that made it easier for us to integrate it with the Habanero environment.

5. There is not enough documentation available on the Habanero API. We had to go through a number of code examples to familiarize ourselves with most of the packages used during the development of our text editor.

6. We unsuccessfully tried to use the swing components [29] for the text-editor. The swing components did not show up when we tried to run our editor within the Habanero framework. Eventually we had to go with the Java's simpler AWT (Abstract Window Toolkit) [30] components to implement our text-editor.

7. The client, when joining a session, opens-up all the applications that are being currently used in that particular session. We feel the user should have control over what applications are initialized upon joining a session. This will enable a user connected at low bandwidth to select only those applications he/she can afford.

Chapter 9

CONCLUSIONS

The results of analysis and experimentation performed on Habanero during this thesis work fully supports our view that frameworks can significantly reduce the time and cost involved in developing quality collaborative applications. Habanero provides a set of functionality that is necessary to accomplish collaboration. Additionally, it also provides simple mechanisms to enable developers to integrate this functionality into their applications. The Habanero framework can be considered as being very close to the black-box architecture. Hence, it does not require the developers to know the insides of the framework in order to plug-in their applications into the Habanero environment. Since Habanero is implemented purely using Java, it is a flexible framework whose functionality can be extended and reused by using object-oriented techniques like inheritance and delegation design patterns [33].

Habanero encourages design as well as code reuse. The concept of code-reuse reduces the development time of an application and design-reuse adds uniformity across all applications developed using the same framework. The result is a set of applications that are easy to develop and maintain. Design-reuse also enables developers to automatically inherit time-tested design-techniques into their applications.

Habanero is a step in the right direction. However, it lacks certain features as discussed during the course of this work. As these features are added, we think Habanero will evolve into a stable platform for the development of real-time collaborative applications. However, it will be pertinent that the future releases of

Habanero are accompanied with a complete documentation of its API. This will help developers to fully exploit the power of Habanero in order to develop enterprise-level collaborative applications, which will also further the cause behind providing such a complex framework.

Habanero uses client/server architecture and replication to achieve collaboration. The information to be shared is distributed by the server, which opens a TCP/IP socket connection to each client to pass on this information. This technique of opening a connection to each client would not scale well in case of large number of participants. However, Habanero provides the ability to use multicasting, which can be used to send data to multiple clients at the same time and hence opening only one connection at a time. But, in order to use multicasting the participants have to be on a router that is connected to the multicast internet backbone (MBone). And, unfortunately not everyone on the Internet is on the MBone.

Habanero uses Java's object-serialization [31] technology to distribute objects. This technology allows the object to be shared by any Java application. However, we believe this distribution of objects can be enhanced if it is achieved by using the XML (Extensible Markup Language) [32] technology. XML enables developers to store data in form of a document. The data in a XML document is represented by a set of user-defined tags, and this data can be extracted by any application, irrespective of the programing language used to implement this application, that has the ability to parse this document. In essence XML makes data portable across platforms, environments and language of implementation. And this provides an enormous opportunity to interface Habanero with external systems.

The power of XML is being rapidly extended by projects like Castor [34], which let developers store Java objects in an XML document and then reconstruct these objects from these documents at a later time. This technology can be integrated within the Habanero framework to add persistence to the objects generated during a session. XML can be further used to provide a standard interface to access data from back-end POS discussed previously.

The Habanero framework only supports applications that developed using Java. However, this should be extended with the use of technologies like CORBA, which enable communication between applications developed using different programming languages.

A tremendous amount of work is being done to raise the level of service provided by the Internet to the data that flows through it. Several protocols have been defined like the RTP (Real-Time Protocol) to achieve real-time communication over the Internet. The future releases of Habanero should try to take advantage of the functionality provided by these protocols. The current release includes an empty class package for these protocols, which demonstrates that Habanero does intend to support these protocols in: the future.

The future-versions of Habanero intend to support collaboration across sessions, where a participant will be able to seamlessly participate in number of sessions at the same time. This ability to simultaneously participate in multiple sessions, the back-end POS support and integration of asynchronous collaboration would create an information-rich environment where collaboration is taking place independent of any time, space or logistic constraints.

When fully developed, ISAAC intends to provide seamless collaboration, which will be independent of time or space constraints. The participants will be able to join a session at anytime from anywhere. By using a POS mechanism the participants of a session will be able to connect to massive amounts of data from which they will easily be able to extract any relevant information.

Habanero is a step in the right direction for achieving true collaboration. It is an excellent framework, which enables developers to build real-time multi-user applications in no time.

REFERENCES

1. [David D Clark, Scott Shenkar, Lixia Zhang] A Service Model for an Integrated services Internet -Internet Draft 1994.
2. [D. Clark, R Braden, S. Shenker] Integrated Services in the Internet Architecture: An Overview .(sunsite.cnlab-switch.ch /ftp /doc/rfc/16xx/163 3).
3. [Domenico Ferrari] The Tenet Experience and the Design of Protocols for Integrated Services: InternetWorks 1994.
4. [Pasquale Di Genova and Giorgio Ventre] Efficiency Comparison of Real-time Transport Protocols 1995.
5. [Paul P. White] RSVP and Integrated Services in the Internet: A Tutorial. May 1997 IEEE Communication Magazine.
6. [Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala] RSVP: A New Resource Reservation Protocol Sept 1993-IEEE Network.
7. [C Partridge] A Proposed Flow Specification, Internet RFC Sept 1992.
8. [Jon Crowcroft, Ian Wakeman, Mark Handley, Stuart Clayman and Paul White] Internetworking Multimedia UCL Press, 1996.
9. Habanero web site: http://havefun.ncsa.uiuc.edu .
10. [Edward L. Peters and M. Pauline Baker]Comparing Middleware Support Systems for Collaborative Visualizations. http://www.ncsa.uiuc.edu/Vis/Publications/collab.html .
11. [Jackson, L] Java Collaborative Technology Selections in NCSA Habanero. Concurrent Engineering in Construction -- Challenges for the New

Millennium, Proceedings of the 2nd International Conference on Concurrent Engineering in Construction, 25-27 August 1999, CIB Publication 236, Espoo, Finland, pp 37-46.

12. [Jackson, L., Grossman, E.] Integration of Synchronous and Asynchronous Collaboration Activities. Computing Surveys, The Association for Computing Machinery, New York, NY, USA, 1999.

13. [Chabert, A., Grossman, E., Jackson, L., Pietrowicz, S., Seguin, C.] Java Object-Sharing in Habanero. Communication of the ACM, The Association for Computing Machinery, New York, NY, USA, Volume 41, Number 6, June 1998, 69p.

14. Synchronous/Asynchronous Blending in Habanero Draft White Paper. http://www.ncsa.uiuc.edu/SDG/Projects/ISAAC/Blending.html .

15. ISAAC web site. http://www.ncsa.uiuc.edu./SDG/Projects/ISAAC .

16. [S.Shenker, C.Partridge and R. Guerin] Specification of Guaranteed Quality of Service, Internet Draft August 1996.

17. [Michael Koch] Design requirements of a Multi-User Text Editor.

18. [Liam J. Bannon] CSCW: An Initial Exploration. Scandinavian Journal of Information Systems, volume 5.

19. [J. Wroclawski] Specification of the Controlled-Load Network Element Service, Internet Draft, August 1996.

20. [Liam J. Bannon] Issues in Computer-Supported Collaborative Learning. Chapter to appear in Proceedings of NATO Advanced Workshop on Computer-Supported Collaborative Learning (Claire O'Malley, Editor) held in Maratea, Italy, Sept. 1989.

21. [Liam J. Bannon] The Context of CSCW. Report of COST14 "CoTech" Working Group 4 (1991-1992), pp. 9-36 Feb 1993.

22. [Greif, I.] Computer-Supported Cooperative Work: A Book of Readings. San Mateo, CA: Morgan Kaufmann, 1988.

23. [Bowers S., Benford] CSCW: Four Characters in Search of a Context* In J. (Eds.) Studies in Computer Supported Cooperative Work: Theory, Practice and Design. Amsterdam North-Holland, 1991. pp 3 -16. (Initially appeared in Proceedings of the First European Conference on CSCW, Sept. 1989, Gatwick, UK.)

24. [Suchman, Lucy A.]. "Office Procedures as Practical Action: Models of Work and System Design," ACM ransactions on Office Information Systems, vol. 1, no. 4, October 1983, pp. 320-328.

25. Groupware: Collaborative Strategies for Corporate LANs and Intranets is now available from Prentice Hall.

26. [Fayad, M.E., Schmidt, D.C] Object-oriented application frameworks, Communications of the ACM, Oct 1997, vol 40, no. 10.

27. [Johnson, Ralph E.] Frameworks = Components + Patterns, Communications of the ACM, Oct 1997, vol 40, no. 10.

28. [Codenie, W.M, Handit Koen D. Steyaort, Patrick, Verccammen, Arlette] From custom applications to domain-specific framework, Communications of the ACM, Oct 1997, vol 40, no. 10.

29. The Swing Connection, http://java.sun.com/products/jfc/tsc/index.html

30. The AWT in Java 1.0 and 1.1, http://java.sun.com/products/jdk/awt

31. Object Serialization, http://java.sun.com/products/jdk/1.1/docs/guide/serialization/

32. [Freter, Todd] XML: Mastering Information on the Web, http://www.sun.com/980310/xml/

33. [Grand, Mark] Patterns in Java, vol 1, Wiley Computer Publishing.

34. [Arkin, Assaf, Visco, Keith] Castor, Java O'Reilly Conference. http://castor.exolab.org/ora-mar-2k/castor.htm .

35. [Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala], RSVP: A New Resource Reservation Protocol Sept 1993-IEEE Network.

36. [Howard, R] "CSCW: What does it mean?" Proceedings of CSCW'88, Portland, September, 1998.

37. DARPA web site. http:// www.darpa.mil/

38. [Johnson, R.E. and Foote, B.] Designing reusable classes, Object-Oriented programming 1, 5June/July, 1998.

39. [Fayad, M.E., Schmidt, D.C, Johnson, R.E] Object-Oriented Application Frameworks: Problems and Perspectives, Wiley, NY, 1997

40. [Edward, J. Posnack, R. Greg Lavender Harric M) An Adaptive Framework for Developing Multimedia Software Components. Communications of the ACM, Oct 1997, vol 40, no. 10.

APPENDIX A- PROGRAM LISTING

TextEditorFrame.java

```
package CollabTextEditor;
import java.awt.*;
import java.io.*;
import java.awt.event.*;
import java.applet.*;
import ncsa.habanero.*;
import ncsa.habanero.streams.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.net.*;
import java.applet.*;

public class TextEditorFrame extends Hablet

{
  int LOCAL_EVENT = 7777;
  public static final int OPEN_FILE_ACTION = 0;
  public static final int SELECT_ACTION = 1;
  public static final int GET_KEY_ACTION = 2;
  public static final int RELEASE_KEY_ACTION = 3;
  public static final int KEY_TYPED_ACTION = 4;
  boolean inited = false;
  boolean sentByMe = false;
  TextEditor TextEditorObject;
  public MirrorFrame mf;
  public SessionParticipant sessionMe;
  public Session session;
  public String name;
  String txt = "";
  public TextEditorFrame()
  {
    super();
    enableEvents(CustomEvent.HAB_EVENT_MASK);
}
public void init()
{
    TextEditorObject = new TextEditor( this, mf );

    add("Center", TextEditorObject);
    TextEditorObject.init();

    //inited = false;
}

public void startInFrame(MirrorFrame f)
{
    this.mf = f;
    Key editorKey = Key.NoKey;
    f.setTitle("Text Editor");
    f.add("Center", this);
    f.setSize(700,500);
    f.addEventCode( Event.ACTION_EVENT );
    f.setLayout(new BorderLayout());
    f.show();
    this.init();
    if (!inited)
    {
        System.out.println("Starting for the firset time");
        EditorLock el = new EditorLock();

        String id = Habanero.id();
        ncsa.habanero.Collobject c = f.getCollobject();
        session = c.getSession();
        sessionMe = session.getParticipant( id );
        name = sessionMe.getName();


        inited = true;
    }
    else
    {
        (TextEditorObject.sharedArea).setText( txt );
        System.out.println("Starting again...");
    }
    validate();
  }


  protected void readHablet(MarshallInputStream in) throws IOException,
                                                   ClassNotFoundException
  {
    TextEditorObject = (TextEditor) in.readObject();
    txt = (String) (TextEditorObject.sharedArea).getText();
    (TextEditorObject.controlArea).releaseLock.setEnabled( false );
    inited = true;
  }

  protected void writeHablet(MarshallOutputStream out) throws IOException
  {
    out.writeObject(TextEditorObject);
  }

  public void sendOpenFile()
  {
    System.out.println("Sending Open File Event " );
    Habanero.sendEvent(   new   CustomEvent(   this,   8889,   (String)
(TextEditorObject.sharedArea).getText(), OPEN_FILE_ACTION) );
    sentByMe = true;
    System.out.println("Finished sending Open File Event " );
  }
  public void sendInsertAction(int pos, String k)
  {
    KeyWrapper obj = new KeyWrapper( pos, k);
    System.out.println("Sending insert text Event " );
    sentByMe = true;
    Habanero.sendEvent(   new   CustomEvent(   this,   8889,   obj,
KEY_TYPED_ACTION) );

    System.out.println("Finished sending Open File Event " );
  }

  public void sendSelectAction(int start_pos, int end_pos, int action)
  {
    ObjectWrapper obj = new ObjectWrapper( start_pos, end_pos, action);
    sentByMe = true;
    Habanero.sendEvent( new CustomEvent( this, 8889, obj, action) );
  }
  public void getKey()
  {
    System.out.println("Sending Open File Event " );
    ((TextEditorObject.generalArea).lockInfo).setText("");
  (TextEditorObject.generalArea).isKeyAvailable = true;
  ((TextEditorObject.sharedArea).txtArea).setEditable( true );

    sentByMe = true;
    Habanero.sendEvent(    new    CustomEvent(    this,    8889,    name,
GET_KEY_ACTION) );

    System.out.println("Finished sending Get KeyEvent " );
  }

  public void releaseKey()
  {
    System.out.println("Sending Release Key Event " );
    (TextEditorObject.generalArea).isKeyAvailable = false;
    ((TextEditorObject.sharedArea).txtArea).setEditable( false );

    sentByMe = true;
    Habanero.sendEvent(    new    CustomEvent(    this,    8889,    "0",
RELEASE_KEY_ACTION) );

    System.out.println("Finished Release Key KeyEvent " );
  }


  public void processEvent( AWTEvent e )
  {
    System.out.println("PROCESSING EVENT");
    if( e instanceof CustomEvent)
    {
      System.out.println("The event was instance of CustomEvent");
      if(! sentByMe )
      {
        CustomEvent he = (CustomEvent) e;
       if( he.getUniqueID() == OPEN_FILE_ACTION )
       {
        (TextEditorObject.sharedArea).setText( (String) he.getArg() );
       }
```

```
          else if( he.getUniqueID() == SELECT_ACTION )
          {
           ObjectWrapper o = (ObjectWrapper) he.getArg();
           (TextEditorObject.sharedArea).select( o.getStartPos(), o.getEndPos()
);

          }
          else if( he.getUniqueID() == GET_KEY_ACTION )
          {
           System.out.println("acting on GET KEy action.");
           (TextEditorObject.generalArea).isKeyAvailable = false;
           ((TextEditorObject.sharedArea).txtArea).setEditable( false );

           String name = (String) he.getArg();
           ((TextEditorObject.generalArea).lockInfo).setText( name + " has the
lock.");
            ((TextEditorObject.controlArea).transfer).setEnabled( false );
          }
          else if( he.getUniqueID() == RELEASE_KEY_ACTION )
          {
           System.out.println("acting on Release KEy action.");
           (TextEditorObject.generalArea).isKeyAvailable = false;
           ((TextEditorObject.sharedArea).txtArea).setEditable( false );
            ((TextEditorObject.controlArea).transfer).setEnabled( true );
          }
          else if( he.getUniqueID() == KEY_TYPED_ACTION )
          {

           System.out.println("acting on Release KEy action.");
           KeyWrapper o = (KeyWrapper) he.getArg();
           ((TextEditorObject.sharedArea).txtArea).setEditable( true );
           ((TextEditorObject.sharedArea).txtArea).setText( o.getKey() );
           if( !TextEditor.hasKey )
             ((TextEditorObject.sharedArea).txtArea).setEditable( false );

          }


        }
        else
        {
          sentByMe = false;
        }
        }

    }

    public void newBounds(int w, int h)
    {
    }


}
```

TextEditor.java

```
package CollabTextEditor;
import java.awt.*;
import java.io.*;
import java.awt.event.*;
import java.applet.*;
import ncsa.habanero.*;
import ncsa.habanero.streams.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.net.*;
import java.applet.*;

public class TextEditor extends Applet implements Marshallable,
WindowListener, ActionListener

{
  public static boolean hasKey = false;
  public SharedArea sharedArea;
  public ControlArea controlArea;
  public GeneralArea generalArea;
  private MirrorFrame mf;

  private TextEditorFrame txtFrame;
  public Frame f;
  private Panel p1, p2;

  public TextEditor(TextEditorFrame txtFrame, MirrorFrame f)
  {
    this.txtFrame = txtFrame;
    this.mf = f;
  }

  public void init()
  {
    sharedArea = new SharedArea(txtFrame, "", 24, 60);
    p2 = new Panel();
    p2.add("Center", sharedArea );
    controlArea = new ControlArea( txtFrame );
    generalArea = new GeneralArea( mf, txtFrame, "", 15, 30 );
```

```
        p1 = new Panel();
        p1.setLayout( new BorderLayout() );
        p1.setBackground( Color.green );
        p1.add("North", controlArea);
        p1.add("Center", generalArea);
        setupEditor();

    }

    public void start()
    {

    }

    public void setupEditor()
    {
        Object parent = this.getParent();
        while(!((Container)parent instanceof Frame))
        {
            parent = ((Container)parent).getParent();
        }
        f = (Frame) parent;
        f.addWindowListener( this );
        f.add("West", p2);
        //f.setLayout( new BorderLayout());
      f.add("East",p1);

    MenuBar mBar = new MenuBar();
        f.setMenuBar( mBar );
        Menu file = new Menu("File");
        Menu format = new Menu("Format");
        mBar.add( file );
        mBar.add( format );
        MenuItem      open      =      new      MenuItem("Open",      new
MenuShortcut(KeyEvent.VK_O));
        open.addActionListener( this );
        open.setActionCommand("open");
        file.add( open );
        MenuItem      save      =      new      MenuItem("Save",      new
MenuShortcut(KeyEvent.VK_S));
        save.addActionListener( this );
        save.setActionCommand("save");
        file.add( save );
    MenuItem      exit      =      new      MenuItem("Exit",      new
MenuShortcut(KeyEvent.VK_E));
    exit.addActionListener( this );
    exit.setActionCommand("exit");
    file.add( exit );
    MenuItem      font      =      new      MenuItem("Font",      new
MenuShortcut(KeyEvent.VK_F));
    font.addActionListener( this );
    font.setActionCommand("font");
    format.add( font );
    MenuItem      color      =      new      MenuItem("Color",      new
MenuShortcut(KeyEvent.VK_C));
    color.addActionListener( this );
    color.setActionCommand("color");
    format.add( color );
  }

  public void openFile()
  {
    FileDialog fd = new FileDialog(f,"Open file", FileDialog.LOAD);
    FileReader in = null;
    File file_read;
    fd.show();
    String dir = fd.getDirectory();
    String file = fd.getFile();
    System.out.println( file +", "+ dir);
    if(( file == null) || (dir == null))
    {
      fd.dispose();
      return;
    }
    try
    {
      file_read = new File(dir, file);
      in = new FileReader(file_read);
      int chars = 0 ;
      int size = (int) file_read.length();
      char[] data = new char[size];
      while( chars < size)
      {
        System.out.println( chars );
        chars += in.read( data, chars, size-chars);
      }
```

```
        sharedArea.setText( new String(data));
    }
    catch( IOException e)
    {
        System.out.println( e );
    }
    fd.dispose();

}

public void saveFile()
{
    FileDialog fd = new FileDialog(f,"Save file", FileDialog.SAVE);
    DataOutputStream out;
    File file_write;
    fd.show();
    String dir = fd.getDirectory();
    String file = fd.getFile();
    try
    {
        file_write = new File(dir, file);
        out = new DataOutputStream( new FileOutputStream( file_write));
        out.writeBytes( sharedArea.getText());
    }
    catch( IOException e)
    {
        System.out.println( e );
    }
    fd.dispose();
}

public void actionPerformed( ActionEvent e)
{
    String cmd = e.getActionCommand();
    System.out.println("The command is : "+ cmd);
    if( cmd.equals("open"))
    {
        openFile();
  txtFrame.sendOpenFile();
    }
    else if( cmd.equals("exit"))
    {
        System.exit(0);
    }
    else if( cmd.equals("save"))
    {
        saveFile();
    }
}

public void windowClosing(WindowEvent e)
{
    f.dispose();
}

public void windowOpened(WindowEvent e)
{
}

public void windowClosed(WindowEvent e)
{
}

public void windowIconified(WindowEvent e)
{
}

public void windowDeiconified(WindowEvent e)
{
}

public void windowActivated(WindowEvent e)
{
}

public void windowDeactivated(WindowEvent e)
{
}

public void newBounds(int w, int h)
{
}

public boolean isActive()
{
    return true;
```

```
}

public void showStatus(String msg)
{
   System.out.println(msg);
}

public void play(URL url, String name)
{
   System.out.println("Unable to play" + name);
}

public void play(URL url)
{
   System.out.println("Unable to play" + url.getFile());
}

public AudioClip getAudioClip(URL url)
{
   return null;
}

public AudioClip getAudioClip(URL url, String name)
{
   return null;
}
public class TextAreaListener implements TextListener
{
   public void textValueChanged( TextEvent e)
   {

      if( e.getSource() != null )
      {
         System.out.println("The VALUE CHANGED------");
      }
      else
      {
         TextArea temp = (TextArea) Habanero.getSource();
         sharedArea.setText( temp.getText() );
      }
   }
}
public class TextAreaKeyListener implements KeyListener
```

```
 {
  public void keyPressed(KeyEvent e)
  {

    try
    {
     if( e.getSource() != null)
     {
     }
     else
     {
        TextArea t = (TextArea) e.getSource();
        System.out.println("----------TExt is:" + t.getText());
        sharedArea.setText( t.getText());
     }
    }
    catch (ClassCastException evt)
    {
    System.out.println("Error finding the class");
       return;
    }
  }
  public void keyTyped( KeyEvent e) { }
  public void keyReleased( KeyEvent e) {}
 }

}
```

Text.java

```
package CollabTextEditor;
import ncsa.habanero.*;
import java.awt.*;
import java.awt.event.*;


public class Text extends TextArea
{
   public TextEditorFrame txtFrame;

   public Text(String txt, int rows, int cols, TextEditorFrame txtFrame)
   {
      super( txt, rows, cols );
      this.txtFrame = txtFrame;
      this.enableEvents(AWTEvent.TEXT_EVENT_MASK);
   }
   public void processTextEvent( TextEvent e)
   {
      System.out.println("Processing Text");
      if( TextEditor.hasKey )
         txtFrame.sendInsertAction(0, getText());
   }
}
```

SharedArea.java

```
package CollabTextEditor;
import java.awt.*;
import java.awt.event.*;
import ncsa.habanero.*;
import ncsa.awt.draw.*;

public class SharedArea extends Panel implements MouseListener
{
  //public TextArea txtArea;
  public Text txtArea;
  public PaletteBox pb;
  private TextEditorFrame txtFrame;
  private String txt;
  private int rows;
  private int cols;

  public class ColorSelectionListener extends MouseAdapter
 {
   public void mouseClicked( MouseEvent e )
   {
     System.out.println("Color Selected" + pb.getSelectedColor());
     txtArea.setColor( pb.getSelectedColor() );
   }
 }

  public SharedArea(TextEditorFrame txtFrame, String txt, int rows, int cols)
  {
    this.txt = txt;
    this.rows = rows;
    this.cols = cols;
    this.txtFrame = txtFrame;

    setup();
  }
  public void setup()
  {
    txtArea = new Text(txt, rows, cols, txtFrame);
    txtArea.addMouseListener(this);
  pb = new PaletteBox(PaletteBox.HORIZONTAL);
  pb.addMouseListener( new ColorSelectionListener());
  setLayout( new BorderLayout());
```

```
    this.add( "North", txtArea );
    this.add( "South", pb );
  }
  public String getText()
  {
    return (String)txtArea.getText();
  }
  public void setText( String text )
  {
    txtArea.setText( text );
  }
  public void select(int start, int end )
  {
    txtArea.select( start, end );
  }
  public void processTextEvent( TextEvent e )
  {
    if( Habanero.isLocalEvent() )
    Habanero.sendEvent( e );

  }

  public synchronized void mouseEntered(MouseEvent e)
  {
    System.out.println("mouseEntered");
  }
  public          synchronized          void          mouseExited(MouseEvent
e){System.out.println("mouseExited");}
  public synchronized void mousePressed(MouseEvent e)
  {
    System.out.println("PRessed");

  }
  public synchronized void mouseReleased(MouseEvent e)
  {
    System.out.println("Released");
    int start_pos = txtArea.getSelectionStart();
    int end_pos = txtArea.getSelectionEnd();
    System.out.println("The selection is : " + start_pos + " to " + end_pos);
    txtFrame.sendSelectAction(          start_pos,          end_pos,
TextEditorFrame.SELECT_ACTION );
  }
  public synchronized void mouseClicked(MouseEvent e)
```

```
    {
      System.out.println("Clicked");
    }

}
```

GeneralArea.java

```
package CollabTextEditor;
import java.awt.*;
import ncsa.habanero.*;
import ncsa.habanero.streams.*;
import java.util.*;


public class GeneralArea extends Panel
{
   public TextArea txtArea;
   private MirrorFrame f;
   public Label title, lockInfo;
   private Choice participants;
   private Panel p1;
   private TextEditorFrame txtFrame;
   private String txt;
   private int rows;
   private int cols;
   public boolean isKeyAvailable = false;
   public GeneralArea(MirrorFrame f, TextEditorFrame txtFrame,String txt, int
rows, int cols)
   {
      this.txt = txt;
      this.rows = rows;
      this.cols = cols;
      this.f = f;
      setup();
   }
   public void setup()
   {
      txtArea = new TextArea(txt, rows, cols);
      title = new Label("    Lock Information    ");
      title.setBackground( Color.yellow );
      lockInfo = new Label("");
      lockInfo.setBackground( Color.white );
      setLayout( new BorderLayout());
      participants = new Choice();
      Collobject cob = f.getCollobject();
      Session session = cob.getSession();
      Enumeration e = session.getParticipants();
      while( e.hasMoreElements())
```

```
        {
           participants.add( ((SessionParticipant)e.nextElement()).getName());
        }
        p1 = new Panel();
        p1.setLayout( new BorderLayout());
        p1.add("North", title);
        p1.add("Center", participants);
        p1.add("South", lockInfo);
        add(  "North", txtArea );
        add( "Center", p1 );
     }

}
```

ControlArea.java

```
package CollabTextEditor;
import java.awt.*;
import java.awt.event.*;
import ncsa.habanero.*;
import ncsa.awt.draw.*;

public class ControlArea extends Panel
{
   public TextEditorFrame txtFrame;
   public Button transfer;
   public Button releaseLock;
   Key editorKey = Key.NoKey;
   PaletteBox p;

   public class GetLockAction implements ActionListener
  {
   public void actionPerformed(ActionEvent ae)
   {
      System.out.println("Get Lock");
      if( Habanero.isLocalEvent() )
      {
         EditorLock el = new EditorLock();
         editorKey    =    txtFrame.mf.getCollobject().arbitrator().getLock(    el,
editorKey);
         System.out.println("The key is:" + editorKey);
         if( editorKey.equals(Key.NoKey))
         {
            return;
         }
         else
         {
            transfer.setEnabled( false );
            releaseLock.setEnabled( true );
            TextEditor.hasKey = true;
            txtFrame.getKey();
         }
      }
   }
}

public class ReleaseLockAction implements ActionListener
{
   public void actionPerformed(ActionEvent ae)
   {
            System.out.println("Release Lock");
            txtFrame.mf.getCollobject().arbitrator().releaseLock( editorKey);
            editorKey = Key.NoKey;
            TextEditor.hasKey = false;
            releaseLock.setEnabled( false );
            transfer.setEnabled( false );
            txtFrame.releaseKey();


   }
}

   public ControlArea(TextEditorFrame txtFrame)
   {
      this.txtFrame = txtFrame;
      this.setBackground( Color.red );
      setup();
   }
   public void setup()
   {
      transfer = new Button(" Get Lock.");
      transfer.addActionListener( new GetLockAction() );
      releaseLock = new Button("Release Lock");
      releaseLock.addActionListener( new ReleaseLockAction() );
      this.add( transfer );
      this.add(releaseLock );
   }
   public void setSize(int height, int width)
   {
      super.setSize( height, width );
   }


}
```

ObjectWrapper.java

```
package CollabTextEditor;

import java.io.*;

public class ObjectWrapper implements Serializable
{
   int start_pos, end_pos;
   int action;

   public ObjectWrapper( int start_pos, int end_pos, int action )
   {
      this.start_pos = start_pos;
      this.end_pos = end_pos;
      this.action = action;
   }
   public int getStartPos()
   {
      return start_pos;
   }
   public int getEndPos()
   {
      return end_pos;
   }
}
```

CustomEvent.java

```
/*
* This class has been modified from HabEvent class which is
* part of mpEdit editor designed and implemented by John
* Jensen.
*/

package CollabTextEditor;
import java.awt.event.*;
import java.awt.*;
import ncsa.habanero.db.*;

public class CustomEvent extends AWTEvent implements StorageMode
{
private static final long serialVersionUID = -3482600173282549405L;
  protected Object arg;
  protected int uniqueID;

  public final static long HAB_EVENT_MASK = 0x4000;
  public final static int OBJECT_SEND = 8889;
  public CustomEvent(Object source, int id, Object arg)
  {
    super(source, id);
    this.arg = arg;
  }

  public CustomEvent(Object source, int id, Object arg, int uniqueID)
  {
    super(source, id);
    this.uniqueID = uniqueID;
    this.arg = arg;
  }

  public Object getArg()
  {
    return arg;
  }

  public int getUniqueID()
  {
    return uniqueID;
  }
  public int getStorageMode()
  {
    char kc;
    KeyEvent ke;
    Object o;

    //System.out.println("In getStorageMode.");

    o = this.getArg();

    if(o instanceof KeyEvent)
    {
      ke = (KeyEvent)this.getArg();
      kc = ke.getKeyChar();

      if (kc == '\n')
      {
        //System.out.println("Storing...");
        return StorageMode.STORE;
      }
      else
      {
        //System.out.println("Accumulating.");
        return StorageMode.ACCUMULATE;
      }
    }
    else
      return StorageMode.STORE;
  }
}
```

VITA AUCTORIS

NAME:	Harvinder S. Minhas
YEAR OF BIRTH	1972
EDUCATIONS:	Shivaji University, India 1990-94 B. Engg (Computer)
	University of Windsor, Windsor, Ontario 1996-2000 M.Sc.