Exploring JSDA, CORBA and HLA based MuTech's for Scalable Televirtual (TVR) Environments

Daniel Dias[1], Geoffrey Fox[2], Wojtek Furmanski[2], Vishal Mehra[1], Balaji Natarajan[2], H.Timucin Ozdemir[2], Shrideep Pallickara[2], Zeynep Ozdemir[2]

[1] IBM T.J. Watson Research Center
[2] Northeast Parallel Architectures Center(NPAC) at Syracuse University

Submitted to the **Workshop on OO and VRML** in the VRML98 Conference to be held at Monterey, California on Feb 16-19,1998.

Abstract

We discuss here new distributed computing technologies of relevance for building multi-user scalable televirtual (TVR) environments on the Internet such as: Java Shared Data API (JSDA) by JavaSoft, Common Object Request Broker Architecture (CORBA) by Object Management Group (OMG) and High Level Architecture (HLA) by Defence Modeling and Simulation Office (DMSO). We describe our early TVR prototype based on VRML2 front-end and JSDA back-end, and we summarize our ongoing work on exploring CORBA Events and HLA Dynamic Data Distribution technologies for building scalable collaboration servers for the Internet.

1 Introduction

Java scripting support in VRML2→VRML97 set the stage for experimenting with multi-user distributed virtual environments on the Internet [1, 2], hereafter referred to as televirtual, or TeleVR, or shortly TVR environments. A typical minimal configuration of such a system would include a few VRML2 browsers, downloading a common VRML world, opening Java node based connections to a collaboratory server which maps user's input such as mouse motions on the suitable movements of the corresponding avatars.

We developed a simple prototype TVR environment of this type at Syracuse University within a joint project [3] with IBM T.J. Watson using JSDA [4] framework for building Java collaboratory services.

Several other prototype TVR environments of similar type were developed recently by various groups [Sony, Paragraph(Mitra), BlackSun, MERL etc.] and a set of VRML SIGs was formed such as Universal Avatars, Humanoid Animation [5] or Living Words [6], focused on standardizing various aspects and software layers of VRML based networked VR.

The detailed architecture of collaboratory servers is not being directly addressed by the VRML community. For example, Living Worlds [6] encapsulates various multi-user technologies in terms of a **MuTech** node and focuses on its interactions with local/client side VRML nodes in the scene graph. There are some associated ongoing standard efforts [7] in the MuTech domain. Recently, Open Community led by Mitsubishi Electric Research Labs (MERL) released an open standard proposal.

In the VRML community framework, we can express our work and the content of this paper as research into promising MuTech technologies that are based on stable open standards and are capable to enable or facilitate the design or development of truly scalable TVR environments.

We are exploring the following collaboratory server technologies of relevance for TVR within the ongoing R&D activities at NPAC:

- JSDA from JavaSoft
- CORBA objects and Event Services
- HLA/RTI by DMSO

In this paper we expose the CORBA domain and discuss relations between VRML and the emergent distributed object technologies.

2 TVR Front-end Description

Our current TVR prototype has two versions of the VRML+Java front-end: one is based on Script Nodes and the other is based on External Authoring Interface(EAI) [8].

EAI version of the prototype was tested on SGI's CosmoPlayer(1.0.2) version running as a plugin to Netscape 3.0 Web Browser on PC platform. Script Node version of the prototype was tested on Sony's Community Place running as a plugin to Netscape 3.0 web browser and also SGI's CosmoPlayer(1.0beta3a) running as a plugin to Netscape 3.0 web browser.

Our current 'world' metaphor is given by a set of rooms with avatars represented by simple geometrical objects (such as colored cones). We are now adding more realism in terms of more humanlike avatars with custom behaviors, conforming to the specifications of the Humanoid Working Group Proposal. We are also exploring add-on audioconferencing capability using commodity API's like Microsoft NetMeeting.

3 TVR Back-end Description

Java Shared Data Architecture (JSDA) [4] provides a Shared Framework for Java at the data level. Data objects are shared over specific instances of Channels (broadcast communication paths) between two or more Clients (objects which are thesource or destination of data) in a collaboration environment. Any Client object, which needs to register its interest in receiving messages sent over a channel, must implement the *Channel Consumer*. In a similar way, if a client is interested in being notified about changes in the state of some other object it should implement the *Channel Observer* interface. To register interest in a certain channel, a client first needs to join the session that hosts this channel and then to join the channel.

JSDA allows to share objects using object serialization mechanisms since it has the Remote Method Invocation (RMI) based implementation. JSDA also has objects which encapsulate management policies for application objects. One example is the Session Manager which authenticates clients to determine if they could join a session.

Currently JSDA is a research-oriented API at JavaSoft Corporation and our TVR prototype is being packaged, as an effective demonstration of JSDA's capabilities, along with the main distribution.

4 Towards multi-server JSDA environments

Figure 1 illustrates a more complex TVR World(currently under development at NPAC) including N avatars in M rooms where both N and M can be large(Internet Clubs, Malls etc). Rooms are mapped to sessions(1,2, etc.) running on individual servers. Each room/session publishes local sensory channel used to exchange coordinate/visual information between avatars in this room. Some rooms can also publish long range channels (e.g. audio) which are accessible from other rooms.

Figure 1 illustrates an avatar moving from room1 to room2. It detaches from room1 visual/sensory channel and attaches to room 2 visual/sensory channel and retains the radio channel to listen news/ads/broadcast from room 1. JSDA Sessions are mapped on "rooms" and JSDA Channels are assigned to individual avatars, present in

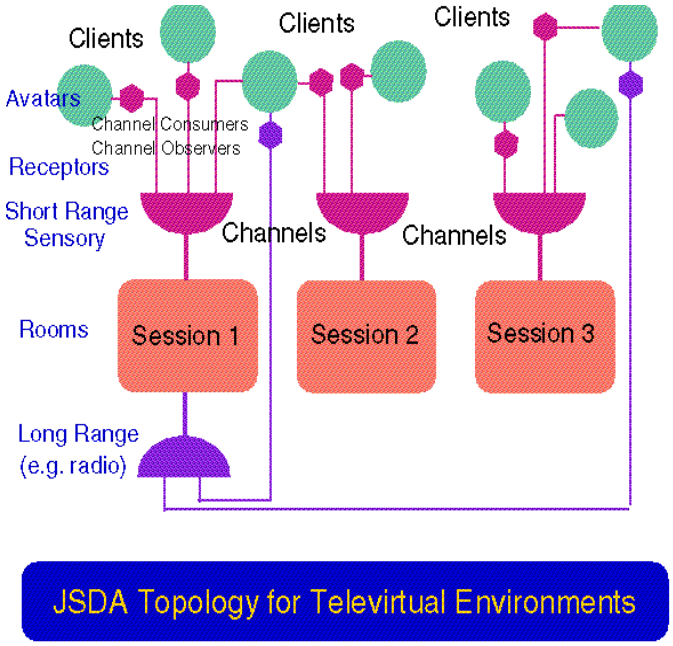

Figure 1: JSDA based TVR

a given room. Only limited number of avatars per room is allowed, and there is also a limit on number of Sessions per collaboratory server. This simple model assures world-wide scalability, assuming that new rooms join with their own session servers and that most interactions are local.

5 Towards CORBA based Collaboratory Environments

JSDA is a useful framework for prototyping simple collaboratory applications but it does not offer either a wire protocol or a high-level API for client-server communication - messages are typically passed as strings, custom encoded/decoded by JSDA clients/servers. The family of T12x protocols (which in fact influenced the JSDA design and was adopted by Microsoft's NetMeeting) could be a natural candidate for a TVR protocol. Another possibility is that such a protocol would be developed as in the course of current interactions between MPEG-4 [9] and VRML Streaming groups. However, another tempting alternative is to select one universal wire protocol for the Internet that would be capable to support all required communication patterns in all relevant application domains.

At the moment, the most promising candidate for such *lingua franca* on the Web is Internet Inter-ORB Operability Protocol (IIOP) by OMG [10] that enables interoperation between ORBs from various vendors and is also frequently used as internal inter-ORB protocol between clients and servers within a single vendor CORBA environment. In the 100% pure Java sector, similar support is offered by RMI (in fact supported as one of the JSDA implementation modes), whereas CORBA offers both multi-platform and multi-language support in terms of the universal IDL interfaces/mappings and language-specific bindings. With the onset of ORBlets, dynamically downloadable or resident in Web browsers such as supported by Netscape/Visigenic, CORBA gets now in fact even more tightly integrated with Java towards a new powerful computing paradigm often referred to as Object Web [11].

Also it is imperative that to operate in today's heterogeneous computing environments, distributed applications must work on a plethora of hardware software platforms. Suitability to business class applications calls for capabilities beyond conventional web based computing - scalability, high availability, performance and data integrity. This is where Java CORBA play a role which mutually complements each other, Java provides for easier distribution of CORBA-based applications with CORBA providing the where-with-all of a distributed infrastructure.

6 Java-CORBA combination

Java's multi-threading support encouraged developers to write web-based distributed software based on proprietary server protocols. Each such server can be viewed as a specific remote computational object. On the other hand, CORBA offers a generic support for such server objects based on distributed object technology. Instead of encoding low level messages, sending them through the network and decoding them at the receiver side, programmer just calls an appropriate high level method on a distributed object without the need of any specific low level network programming. This high-level abstraction capability is definitely a promising framework for the future distributed solutions. The only two alternatives that can be viewed as competivtive are Java RMI and Microsoft DCOM - but only CORBA is both language- and platform-independent.

However, rather than a competition or alternative to CORBA, Java is being now viewed by many as a complementary technology which forms a perfect match with CORBA within the emergent 'Object Web' trends. In a nutshell, the master plan of the Object Web (supported by Netscape, Oracle, IBM, Sun and others) is to implement CORBA control i.e. the middleware layer (including ORBs and some core services) in Java.

Since Java has an inverse mapping to IDL, a programmer can stay in the Java environment during the software development. Java-CORBA implementations can run on thin network computers and low-end consumer devices because of their low-complexity and footprint. Java's mobile byte code and CORBA's Dynamic Invocation Interface (DII) simplifies upgrades of clients' software in large distributed systems. Java and CORBA combination truly provide the right building blocks for a distributed object computing such as: a) platform-independence, strong security model etc. in Java language; and b) static and dynamic interfaces, synchronous and asynchronous method calls with the comprehensive set of Facilities and Services in CORBA.

These factors might result in the CORBA/Java combination to assume a central role in shaping the Internet during the next phase of its evolution. Such emergent Object Web could have impact in several areas, including multi-user collaboratory environments.

In particular, the collaboratory environments can be naturally addressed by CORBA in terms of the Event Service - one of the standard 15 services developed and sustained by OMG (together with Security, Persistence, Concurrency, Naming, LifeCycle, Relationships, Trading and other such fundamental object services). In the following Sections, we describe the CORBA Event Service which offers similar functionality as the JSDA Channel discussed in Sections 3 and 4.

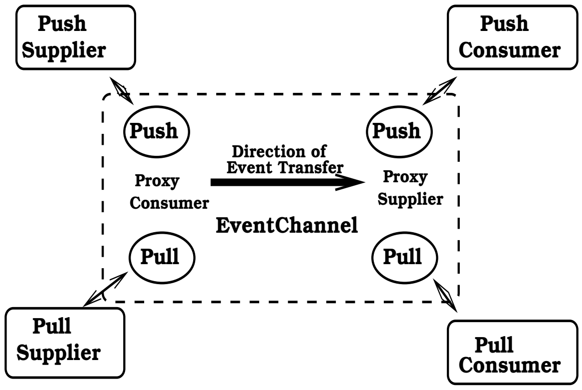

Figure 2: CORBA Event Service

7 CORBA Event Service

The Event Service (ES) allows for decoupled communication between objects: instead of a client directly invoking operation on a target object, it can send an event that can be received by any number of objects. The sender of an event is called **supplier**, and the receivers are called **consumers**. Suppliers and consumers are decoupled i.e. they do not know each other's identities.

The Event Service introduces the notion of Event Channel. Suppliers send events to an Event Channel, and consumers receive these events. Each channel can have multiple consumers and suppliers, and all events sent by a supplier are made available to all consumers of that channel.

The ES supports four different modes of consumer-supplier interactions. The consumer could be push/pull-Consumer, and the supplier could be a pull/push Supplier. One of the advantage of using the EventChannel is that, the events can be buffered to accommodate consumers of differing speeds. The suppliers and the consumers both register with the EventChannel since otherwise its not possible to determine the source of the event in case of the supplier and also since its not possible to invoke the appropriate notification method on the consumer.

Supplier objects ask a proper ProxyConsumer object from the Event Channel's ConsumerAdmin object. Whenever the supplier wants to send an object to the Event Channel, it uses its corresponding ProxyConsumer object. Similarly, consumer objects ask a proper ProxySupplier object from the Event Channel's SupplierAdmin object. Whenever a channel receives an event, it informs all the ProxySupplier objects. Then, each proxy object notifies its consumer.

8 CORBA-Based Collaboratory

Our work on CORBA based collaboratories was initiated at the IBM T.J. Watson Research Facility and is being pursued further through a joint-project [3] with the Northeast Parallel Architectures Center(NPAC), Syracuse University.

So far, we developed an initial API for CORBA based collaboration. The IDL definition of this API is given in the following section. A prototype version of this API using CORBA objects as "shared data" and CORBA servers as "collaboratory servers" and Netscape/Visigenic based ORBlet front-ends has been developed at IBM.

We are currently extending this design and preparing a refined implementation using CORBA Event Service that plays a similar event filtering role as the JSDA Channels.

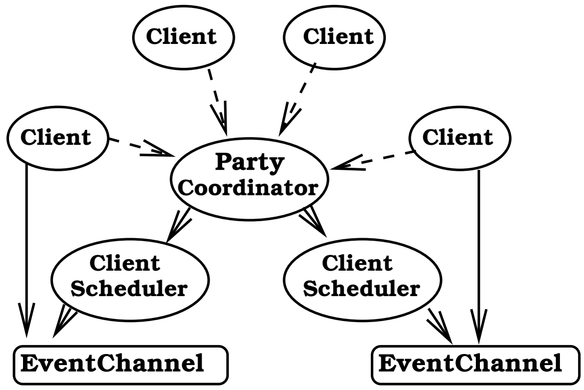

Figure 3: Event Service based Collaboration Framework

These are the two most significant IDL definitions in the Collaborative System. The IDL definitions signify the operations a Client could invoke on a remote instance of these objects. Nevertheless, an invocation of any of these aforementioned operations should be preceeded by a successful reception of a remote handle to these objects. Acquisition of a handle to the PartyCoordinator, requires the client to invoke a bind to that Object. To digress on the semantics of the bind , it should be clear that in a HighAvailability scenario there would be multiple instances of the PartyScheduler with a static Hashtable containing the list of updated Parties i.e. Coordinator Objects.

```
interface Coordinator {
   boolean setMaxClients(in long arg0);
   long getMaxClients();
   long numberOfMembers();
   typedef sequence string sequence_of_string;
   MultiCoordinator::Coordinator::sequence_of_string
             getClientNames();
   boolean isEmpty();
   long register(in long arg0, in string arg1,
             in Client::ClientControl arg2);
   boolean deregister(in long arg0);
   boolean broadcast(in string arg0);
   boolean whisper(in string arg0, in long arg1);
};

interface PartyScheduler {
   boolean createParty(in string arg0);
   long getPartyID(in string arg0);
   MultiCoordinator::Coordinator
             getPartyHandle(in long arg0);

};
```

Elucidating further on the semantics of operations on these remote objects, the PartyScheduler is the one which schedules the appropriate instance of the Coordinator Object to coordinate Clients logged onto a specific session (Party) comprising of possible different applications. Basically, the PartyScheduler is responsible for spawning instances of the Coordinator, possibly across a different subnet, and also for returning an remote Coordinator-handle to the Client. A brief description of the sequence of operations in the Collaborative System follows.

The Client initiates a bind to the PartyScheduler Object. Given that this is successful, in the event that there is a Distributed Directory service and the Active Object server is in place, the Client is now ready to invoke the IDL-defined operations.

1. It starts with the createParty(String partyName) function which would return a true in the event that a new Coordinator Object has been instantiated or a false to signify the prior existence of the desired party. It is the Scheduler's job to signal the appropriate notification to the Clients and perform appropriate housekeeping to reflect new instances of Coordinator's. All Coordinator objects scheduled by the PartyScheduler are identified by an ID.

2. The Client now has the option to decide wether he wishes to join an existing Party or initiate the existence of a new one. In the latter case Step[1] is repeated as mentioned earlier. Once the process is over, the Client gets a handle to the Coordinator Object by invoking long getPartyID(in string arg0); MultiCoordinator::Coordinator getPartyHandle(in long arg0); in succession. This is in keeping with the policy of the PartyScheduler to identify Coordinator's on the basis of the ID that it assigns during their instantiation.

3. Once Steps I and II are over and done with, the Client in a Distributed Collaboration mode, and can invoke operations specified in IDL definitions for the Coordinator. These include boolean broadcast(in string arg0),boolean whisper(in string arg0, in long arg1); among other functionalities offered by the PartyCoordinator Object.

This is just another demonstration of the complementary roles Java CORBA play in a distributed environments. Java provides for easier distribution of CORBA-based applications with CORBA providing the where-with-all of a distributed infrastructure. In summary, CORBA offers both a potential candidate for universal wire protocol, IIOP, and a natural collaboratory framework based on shared CORBA objects and flexible message filtering mechanisms offered by the Event Service.

9 Scalability and Fault-tolerance in Collaborative Systems

Scalability and Fault-tolerance are the essential required features in Collaborative Systems and they can be naturally addressed in the CORBA model.

The replication of servers on different participating hosts in a collaborative environment answers the scalability problem when the load (participating sessions) on a server crosses a certain threshold. The current API supports two different approaches. The notion of groups within a certain session allows us to define one object for each group and place these objects on different machines easily. It is also possible to split the Event Channel if it exceeds the certain capacity and connect two Event Channels to each other as supplier and consumer since one Event Channel can be a consumer/supplier of another Event Channel.

Fault-tolerance in Collaborative Systems can be solved by migrating sessions to a different participating host with minimal or little disruption whenever the machine hosting the server crashes. The ObjectServices agent, which is a distributed directory service, allows for migration of sessions to a different participating host in the case that a session terminates unexpectedly on one of the hosts. The Events can be stored persistently by the Event Channel to ensure that events are not lost on system failures.

10 Using CORBA Event Service for Message Broadcasting

At NPAC, we implemented recently the Event Service for omniORB2 [24], which is a free C++ Object Request Broker under development by Olivetti and Oracle Research Labs. We wrote the standard Event Service with C++ and omniThread thread library. We tested this software with a Chat program using the Netscape 3.0 with OrbixWeb based

ORBlets.

We also start exploring the issues, related to using CORBA Event Service for Distributed Interactive Simulation (DIS) [14, 15] PDU broadcasting and for HLA/RTI support. Some additional services are required to support event handling in multi-user environment. For example, Event Service does not care about the originator of the event. But for multi user environment we need to know who sent the message. DIS PDU format includes this information in its own body. The Event Service is central server based approach compared the peer-to-peer architectures. We can solve this problem by providing an Event Channel for each active object in the virtual environment.

For the large scale interactive simulations, it is imperative that we support event filtering to adjust the frequency of events received from the supplier. The obvious choice is to make this decision a responsibility of Event Channel so that the messages are being handled before they are on the network. Event filtering could be based on time stamping. However, this simple solution has a profound problem with synchronization of time values in distributed simulations. The well-known solution is to provide the Global Virtual Time (GVT) calculation to the system so that out of order messages can be handled properly with the rollback mechanism. GVT calculation allows us to release the storage for the logged events since nobody expects to receive an event time stamped earlier than the current GVT.

One another intriguing option for message transfer is to use multicasting. This requires some changes in the Event Service implementation. For Push Consumer, instead of giving a separate Push Supplier for each consumer, it is possible to give one Push Supplier for each multicast group so that multicast Push Supplier can serve multiple consumers. This change also reduces the computation requirement on the Event Channel server.

Several new advanced event handling features are currently in the OMG standardization pipeline in the form of the CORBA Notification Service [25]. New capabilities include: support for Quality of Service, integration with Transaction and Security Services, and more flexible/user-adjustable format for event objects.

11 Emerging Collaboratory Server Technologies based on Distributed Object Technology

Experiments with Java and CORBA based collaboratories described above represent our first initial steps towards systematic Object Web support for the High level Architecture (HLA) based modeling and simulations.

HLA [16, 17, 18] is a next generation framework [21] for distributed simulation systems and promoted by DMSO (Defense Modeling and Simulation Office) to replace the current DIS standard. HLA's enabling middleware called Runtime Infrastructure (RTI) is based on distributed object technologies and DMSO is promoting HLA/RTI within the OMG towards a Vertical CORBA Facility for the Interactive Modeling and Simulation.

At NPAC, we are working with the DoD High Performance Modernization program on integrating advanced web/commodity technologies with large scale Forces Modeling and Simulation systems, being converted to or already based on HLA by DMSO and the enabling RTI middleware.

As part of this project, we are building: a) an Object Web based implementation of IIOP and HTTP server called JWORB (Java Web Object Request Broker); and b) the Object Web based RTI layer to operateon top of JWORB that will provide Web based Simulation support for HLA and a natural linkage to front-end technologies such as VRML. For large geographically distributed MS systems, middleware must be given by a mesh of scalable collaboratory servers running on heterogeneous platforms and supporting specific simulation components written in various languages. A Java/CORBA based RTI middleware such as JWORB with VRML front-end seems to offer an attractive pervasive architecture for such systems.

Of a particular interest within the DoD MS is the Simulation Based Acquisition or Virtual Prototyping Environments where new systems are engineered and tested in the virtual space before the first real prototype is manufactured.

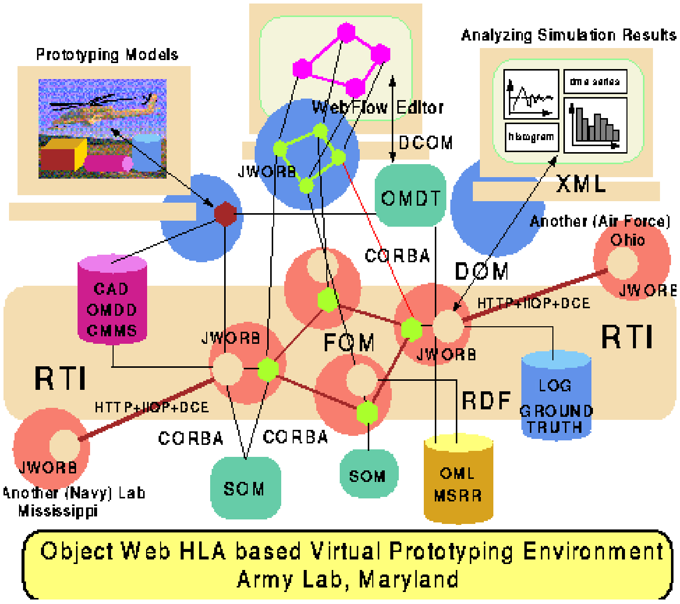

Figure 4: Framework for Virtual Prototyping Environments

Figure 4 illustrates a sample of such a system with JWORB based middleware and a collection of front-ends, including (XML based) data analysis, (Java based) data flow visual authoring software and (VRML based) visual 3D display. Each of these activities can be made collaborative via the base RTI mechanism or via CORBA Event Service or via JSDA, and they can all cooperate via the JWORB based componentware.

12 Summary

We discussed here a set of new promising distributed computing frameworks (JSDA, CORBA, RTI) which offer open standards based support for building scalable multi-user virtual environments on the Internet. The essential feature of such environment - communication locality - is enabled via event filtering in terms of JSDA channels, CORBA Event service and RTI routing spaces [19, 22].

So far, we acquired few early prototyping experience using JSDA and CORBA technologies and we are now exploring the HLA/RTI environment. In our JWORB middleware framework under development, we will be able to integrate, experiment with and conduct comparative analysis of all three collaboratory technologies discussed here: JSDA, CORBA and HLA/RTI.

References

[1] Bernie Roehl, Justin Couch, Cindy Reed, Tim Rohaly and Geoff Brown, *Late Night VRML 2.0 with Java*, 1996

[2] Rodger Lea,Ken Miyashita and Kouichi Matsuda, *Java for 3D and VRML worlds*, 1996

[3] Syracuse University/IBM Technical Report on *Prototype for Scaleable TeleVirtual Environments for the Web*

[4] Java Shared Data Architecture(JSDA) - A collaborative framework under development at JavaSoft. at http://java.sun.com/people/richb/jsda/

[5] VRML Humanoid Animation Working Group at http://ece.uwaterloo.ca:80/ h-anim/

[6] Living Worlds Standards proposals - Mitra, Mitra Internet Consulting. at http://www.livingworlds.com/

[7] John W. Barrus, Richard C.Waters, and David B. Anderson - MERL Research Lab, *Locales and Beacons: Efficient and Precise support for Large Multi-User Environments*, IEEE Computer Graphics and Applications, 16(6):50-57, November 1996, also at http://www.merl.com/reports/TR95-16a/locales.html

[8] External Authoring Interface Reference at http://vrml.sgi.com/moving-worlds/ spec/ExternalInterface.html

[9] Moving Picture Experts Group (MPEG) at http://drogo.cselt.stet.it/mpeg/

[10] CORBA 2.0 Specifications - Object Management Group http://www.omg.org

[11] Dan Harkey and Robert Orfali, *Client/Server Programming in Java and CORBA*, John Wiley Sons, Inc., 1997

[12] Sean Baker, *CORBA Distributed Objects*, Addison-Wesley and ACM Press, 1997

[13] CORBA Services Specifications - Object Management Group http://www.omg.org

[14] *The DIS Vision, A Map to the Future of Distributed Simulation*, 1993, at SISO's DIS section (http://siso.sc.ist.ucf.edu/dis/index.htm)

[15] *IEEE Standard for Distributed Interactive Simulation - Application Protocols*, IEEE 1278.1-1995

[16] *High Level Architecture Federation Development and Execution Process (FEDEP) Model Version 1.0*, at http://www.dmso.mil/projects/hla

[17] *High Level Architecture Interface Specification Version 1.2 -Draft 6*,
at http://www.dmso.mil/projects/hla

[18] *High Level Architecture Time Management Design Document Version 1.0*
at http://www.dmso.mil/projects/hla

[19] Danny Cohen and Andreas Kemkes, *Using DDM - an Application Perspective* , 1997 Spring Simulation Interoperability Workshop (SIW), 97S-SIW-014

[20] James O. Calvin and Richard Weatherly, *An Introduction to the High Level Architecture(HLA) Runtime Infrastructure(RTI)* , March 1996, 14th DIS Wokshop , 96-14-103

[21] Duncan C. Miller, *The DOD High Level Architecture and The Next Generation of DIS* , March 1996, 14th DIS Wokshop, 96-14-115

[22] Katherine L. Morse, *Interest Management in Large-Scale Distributed Simulations* , UC Irvine, Information and Computer Science Technical Report, ICS-TR-96-27 at
http://jblevins.ics.uci.edu/Dienst/UI/2.0/
Describe/ncstrl.uci%2fICS-TR-96-27

[23] DIS-Java-VRML Working Group (headed by Don Brutzman) proceedings.
at http://www.stl.nps.navy.mil/dis-java-vrml/

[24] omniORB2 at http://www.orl.co.uk/omniORB/

[25] CORBA Notification Service Proposals - OMG
http://www.omg.org/library/schedule/
NOTIFICATION_SERVICE_RFP.htm