Web Technologies for Collaborative Visualization and Simulation

Lukasz Beca Gang Cheng Geoffrey C. Fox Tomasz Jurga
Konrad Olszewski Marek Podgorny Piotr Sokolowski Krzysztof Walczak[*]

Abstract

Web technologies—in particular linked Java servers and clients—allow new dynamic collaborative environments linking people and computers. We describe the architecture of a system, TANGOsim, that combines a Java collaborative environment with an executive providing general message filters, and an event driven simulator. The initial application is to command and control, but we describe how this approach can also be used in other areas, such as health care, scientific visualization, and (distance) education.

1 Introduction

We believe that current Web technology offers remarkable opportunities for sophisticated new environments linking people and computers (collaboratories). This includes traditional video conferencing, telemedicine, command and control, crisis management, distance education, computational steering and visualization, and "human in the loop" distributed discrete event simulations. The basic technology is a distributed set of Java servers and clients that communicate. The Java clients are applets launched from traditional browsers. The cited environments can be implemented as a set of communicating processes with the servers coordinating traffic between the clients. In a traditional collaborative environment, the servers run session managers that log new users, and ensure a common world view so that applications spawned on one client (white boards, lessons for distance education, etc.) are shared on all others. This simple sharing of information (Figure 1a) is already a powerful environment, but we can generalize this by allowing client messages to be processed by arbitrary filters before they are passed on to other clients (Figure 1b). This allows different users to receive different views of the same data—in a complex command and control environment. It is clearly necessary to present the different decision makers with the different detail needed for their responsibility. Further, we can drive these filters by dynamic scripts that allow one to include simulated users for training and modelling applications. Note that the "users" (more accurately, client) generating and receiving messages can be either people or computers. For instance computational steering in its simplest case corresponds to a single person interacting with a single computer.

The overall system is now a set of Java servers implementing a (distributed) event driven simulator that accepts events from either scripts or clients corresponding to computers or people.

In this paper, we describe in Section 2 the design and implementation of our experimental system TANGOsim, shown in Figure 2, which has successfully implemented the above concept.

[*]Syracuse University, Northeast Parallel Architectures Center, 111 College Place, Syracuse, New York 13244, gcf@npac.syr.edu, http://www.npac.syr.edu

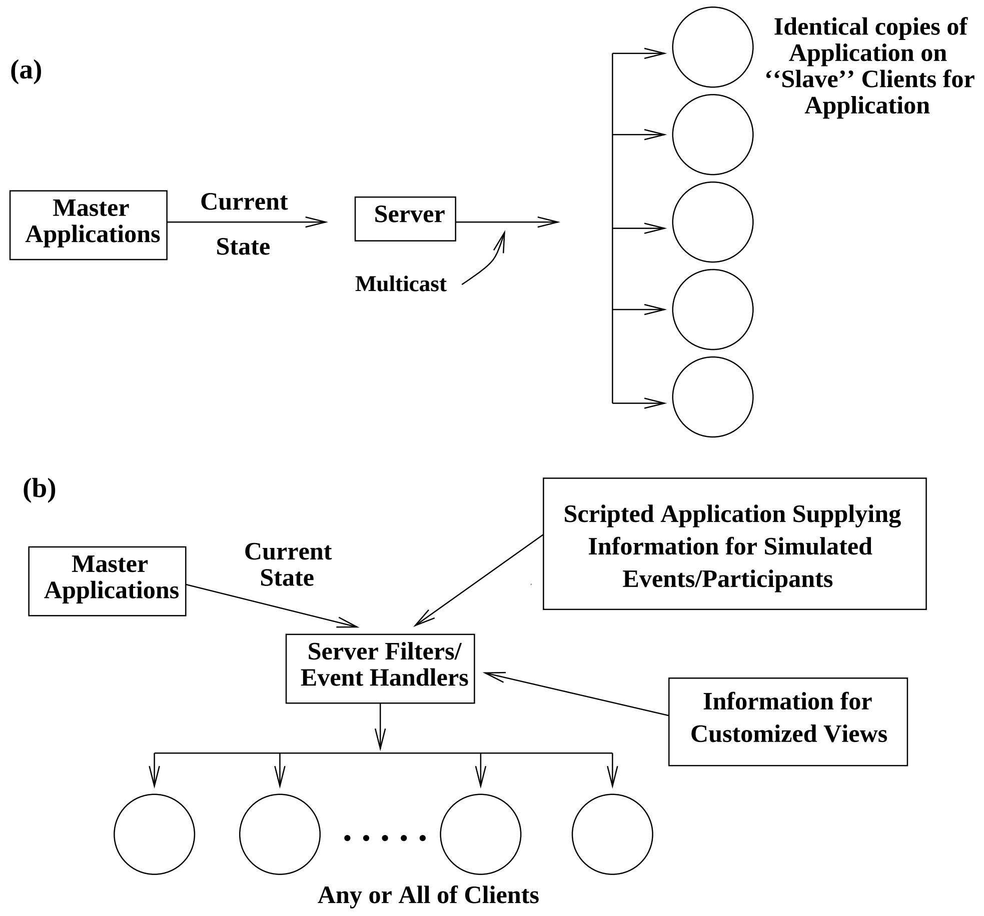

(a)
Master Applications
Current
State
Server
Multicast
Identical copies of Application on ''Slave'' Clients for Application
(b)
Master Applications
Current State
Scripted Application Supplying Information for Simulated Events/Participants
Server Filters/ Event Handlers
Information for Customized Views
.....
Any or All of Clients

Figure 1: (a) Simple TANGO Shared Synchronous Application; (b) Scripted Customized Application

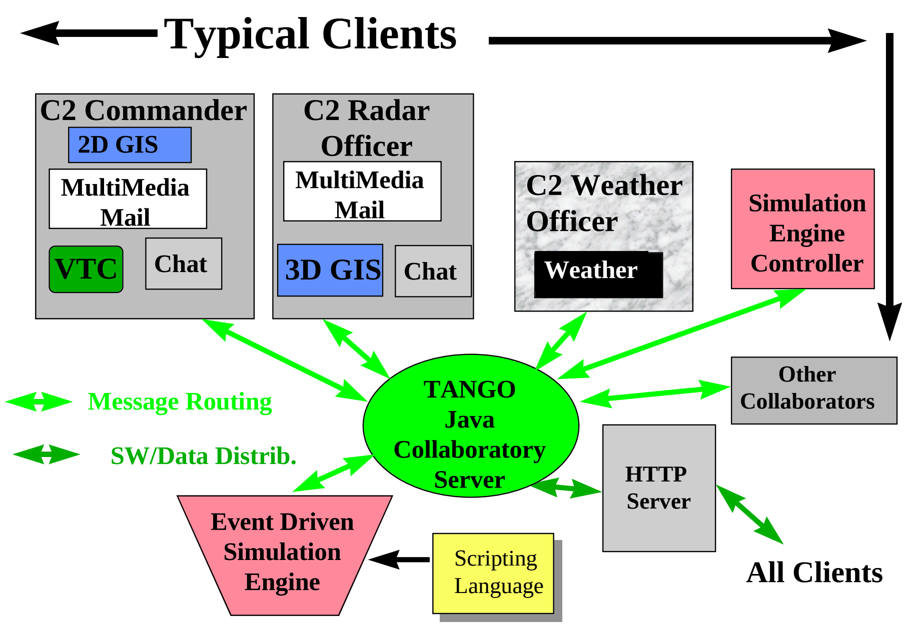


Figure 2: The Command and Control Tango Application

We have already used it in one significant command and control application, described in Section 3, and illustrated in Figure 2.

In Section 4, we speculate on its use in other areas including computational steering, distance education, and health care.

2 TANGO: Design Goals, Architecture, and Implementation

2.1 Rationale

In spite of its wide acceptance, the traditional World Wide Web is little more than a convenient method of accessing and displaying information. Use of the stateless HTTP protocol greatly contributed to the success of the Web, but it also strongly limited its further development. The basic WWW model does not support complex models for interaction. The introduction of the CGI (Common Gateway Interface) mechanism did not solve the problem. At this stage of the World-Wide Web development, it was impossible to create tools for cooperative work based on this technology. The situation has changed with the introduction of some new technologies.

In the 1995 Sun Microsystems, Inc. introduced Java [Gosling:96]. Among other well published features, Java introduces a notion of applets. Applets are programs that can be included into HTML pages, much like an image. A Java-compatible browser can be seen as an extensible container capable of securely executing chunks of the code distributed to the clients from an HTTP server. It is interesting to note that the idea of downloadable interpreted software is hardly new: in the late 80s, Sun introduced a product called NeWS [Gosling:89a] which used extended Postscript as both programming language and communication protocol. The idea was apparently too far ahead of its time—NeWS project was promptly abandoned. The combination of Java and HTTP seems to carry much greater appeal. Java gained enormous popularity among the World Wide Web users. Together with its client side scripted partner, JavaScript, it is now broadly used to bring interactivity into the Web environment.

The notion of interactivity is somewhat fuzzy. Currently, an "interactive" Web application is the one that creates dynamic HTML pages dependent on user input. In this paper we are not concerned with this sort of interactivity. Rather, we describe a system that will allow human users to interact via their computers. Such systems are known under the newspeak term of *collaboratory.*

Computer mediated collaboratory systems have a relatively long history (for an extensive review, see [Schooler:96]). Despite of the extensive body of research, collaboratory systems are hardly out of the laboratory. Academic implementations have been evolving together with the enabling multimedia and networking technologies: only recently the sophistication of the affordable desktop technology reached the level making multimedia collaboratory a viable prospect. The commercial collaboratory implementations are exceedingly primitive: video teleconferencing, shared whiteboard with limited graphical capability, and a textual chat tool represent industrial state of the art, as exemplified by Insoft's Communique!, SGI's InPerson, or Intel's ProShare products. It is not very surprising that this type of collaboratory is perceived by some as an activity in search of a need [Grudin:90].

Why a Web-based collaboratory system? From a sociological standpoint, we believe that today's web is a prototype of the most common environment in which people interact with computers. If computer collaboratory is to become a commonplace, it must be Web based per definition. The overwhelming success of the Web creates new opportunities for collaboratory. From the technological standpoint, collaboratory systems critically depend on interoperability. Web is undoubtedly the most successful implementation ever of an open system. Web's very nature creates a coherent programming environment which is conducive for collaboratory experiments. Java, with its platform independence, is an ideal language for implementation of collaboratory systems.

Another important argument for a web-based collaboratory is the vast information contents of the Web. The limited success of the traditional collaboratory tools is at least partly attributable to their poor integration with information retrieval tools. With little content, collaboratory process becomes meaningless and artificial. Web collaboratory has a potential for seamless integration of information retrieval and exchange tools, with vast data repositories readily available. These conclusions were reached in context of *Telemedicine* by Balch and Warner, who suggested an information rich Bridge System to replace traditional video conferencing based approaches [Warner:95a].

We have designed and implemented a web-based collaboratory system, code-named TANGO [TANGO:96a], which is optimized for system flexibility and the capability of integrating existing applications [Beca:97a]. We use the following design principles:

- We do not really know how to build an efficient collaboratory.

  Since no truly successful collaboratory system exists, there is no blueprint for success. We have to therefore allow for an extensible system with very few limitations. TANGO must not define application specific protocols, application programming language, or limit in whatever way functionality of collaboratory applications.

- The essence of the collaboratory function must be defined by application and by application only.

  It is up to application developer to specify requested collaboratory functionality for every application. This functionality may be obvious, as for a chat or audio conferencing, or highly non-trivial, as for a collaborative weather prediction and analysis application with real-time computational backend. TANGO supports collaborative process for either type of application. Note, we can expect each type of collaboratory to need different capabilities in TANGO.

- The most likely collaboratory applications will be built around existing complex software systems.

  This implies a necessity for a simple API enabling fast and easy porting process of the existing applications into the collaborative framework. TANGO supports such an API for applications written in few languages. By providing such an API, TANGO fully benefits from the following simple observations:

  1. the majority of the code of existing applications may be reused while building their collaborative versions
  2. the majority of applications working in collaborative mode may be implemented using similar and actually simple message exchange mechanism
  3. very useful distributed collaborative systems can be built by integration of slightly modified standalone applications together with some user and session management mechanisms

TANGO is not the only web-based collaboratory system. There are multiple ongoing projects in the area of Web based distributed systems. Some of them aim at development of a platform for implementation of distributed software in Java [NCSA:96]. Some of the others build software simplifying development of such distributed applications [Chandy:97a], [CIP:96]. The majority of other projects concentrate on development of single-purpose distributed or collaborative systems themselves. In designing TANGO, we tried to build a generalized, extensible, Web based collaboratory system capable of creation of diverse shareable information spaces. The system provides support for coordinated but independent views of related information streams, with capability for high-end visualization and interactive manipulation of multimedia information. TANGO extends the Web paradigm to the domain of collaborative computing and well beyond the concept of the chat, shared whiteboard, and replicated, identical instances of simple generic applications. TANGO is a generic tool used to share views.

2.2 System Requirements

We have based design and implementation of the TANGO system on the following requirements:

- Integrate both standalone and Web-based applications by providing a uniform interface to allow them to communicate with their instances on remote machines;

- Allow one to execute and control collaborative application from the Internet browser environment;
- Provide means for session control (user authentication, starting and ending sessions, tracing participants activities, changing user privileges);
- Enable integration of existing applications written in any programming language, assuming socket communication as the only necessary communication mechanism;
- Provide ability to download certain applications across the network through the use of Java applet technology, allowing main parts of the system to be automatically distributed across the Internet;
- Provide logging mechanism, so all user activities may be stored in the persistent form in a database and retraced if necessary;
- Support definition of compatible message and application classes to enable multiple, task oriented views of the information streams

2.3 TANGO System Overview

The main functionality provided by the system consists of the following elements:

- session management

  Each shared application defines a session that has a single *master* user—which status is transferable.
- communication between collaborating applications (data and event distribution)

  There are *control* messages that are invisible to user, and *application* messages that are means by which user applications exchange state and that are not interpreted by system (unless they interact via TANGOsim filters).
- user authentication and authorization
- event logging in back-end database

2.3.1 Session Management

A *session* is a group of application instances currently working together in collaborative mode. All applications belonging to the same session exchange information and share behavior. How a particular application operates in collaborative mode depends on this application characteristics.

Each application belongs to a session, even if it is not currently used for collaboration. In such case the session consists of this one application only.

To provide for the floor control, for each application in a sessions there is one distinguished user which is considered to be a *master*. The master of the session has special privileges of controlling the application behavior and/or controlling access of other users to this session. The privileges depend on the application type. The master status is transferable

For multimedia applications using high-volume data streams (audio/video, hi-res images, 3D), session management interacts with the system layers controlling multicast capability.

2.3.2 Communication

Only applications belonging to the same session can communicate. A robust message passing system supports this communication. Message and event distribution proceed automatically under control of the session manager.

There are two major types of messages: control messages, and application messages.

Control messages are generated by the system for communication among server, daemons, and control applications. These messages serve for logging users into the system, establishing sessions, launching applications, etc. Control messages are invisible to user applications.

Application messages are main means of communication among user applications. The structure of application messages depends on the application and is not interpreted by the communication system. The communication system is transparent to the application messages.

2.3.3 User Authentication and Authorization

System provides means for a multilevel user authentication and authorization. TANGO security mechanisms are independent on the underlying operating system.

2.3.4 Event Logging

One of the important system capabilities is recording of the system activity. Since all system and application messages must go through the main server, all of them can be recorded in a database. Each time a control or application specific message passes the server this fact is recorded in the database together with the date, exact time, and sender information. These data can be then accessed and the whole system activity can be replayed.

Event logging is a basic feature supporting *asynchronous collaboration.* Note that the term asynchronous collaboration is frequently used to describe e-mail and other messaging systems. In the collaboratory context, asynchronous collaboration denotes ability of the collaboratory system to replay and even summarize at a desired level the entire collaboratory sessions.

2.4 System Architecture

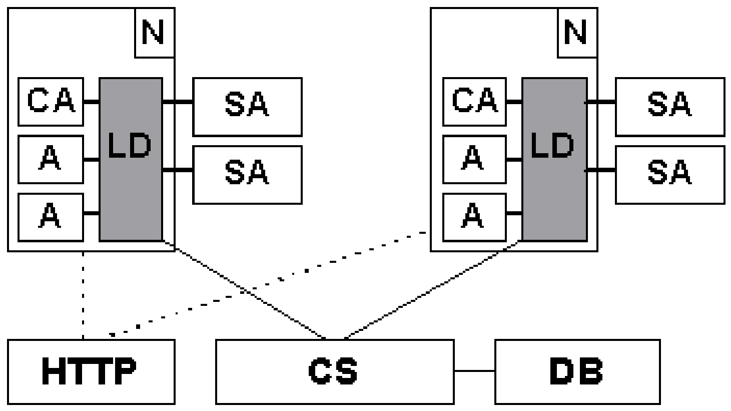
2.4.1 System Elements

Figure 3 presents the global architecture of the system. The system consists of the following components:

- *Local daemon's* main task is to maintain two way communication between user applications, applets and the central server and to lunch local applications. Demon is implemented as a plug-in for the Netscape browser.

- *Central server* is the main communication element. All local daemons communicate with the central server. The server maintains the system state data. Its main tasks are routing of messages among applications participating in each session and recording all events in a database.

All user applications which run as standalone programs are call *local applications.* Local application may be written in any programming language. All local applications communicate



CA	- Control Application
A	- Java Applet
SA	- Standalone Application
LD	- Local Demon
CS	- Central Server
HTTP	- HTTP Server
DB	- Database
N	- Netscape Navigator

Figure 3: Global Architecture of the TANGO System

with the local daemon by the use of sockets. The daemon is responsible for starting these applications and routing messages to and from applications.

Java applets are also user applications but written in Java language, downloaded through the network from an HTTP server, and executed in Netscape environment. Communication between Java applets and central server is also maintained by the local daemons. Java applets communicate with local daemon by calling its method functions.

Control application is a specialized application which acts as an interface to the user. The control application is used to launch applications locally or remotely, create and join existing sessions, exit applications, etc. This application allows also logging into the system. User interface to the control application depends on his/her privileges, giving a user access to only these features that he/she is allowed to use. The control application is customizable.

2.4.2 TANGOsim Extension

An example of TANGO architecture flexibility and extensibility is TANGOsim—an discrete event simulator. A multithreaded simulation engine implementing virtual time can be driven by either a scripting language or interactively controlled by a user via Simulation Controller. The engine and the controller are implemented as a Java application and a Java applet, respectively. The simulation engine can create messages for any application compatible with TANGO system, to create and control sessions, and to realize scenarios in which the course of action depends on user input. TANGOsim is a prime example of the system capability to go beyond the collaboratory model of "cooperating twins." TANGOsim also supports filtering messages, and linkage of messages between sessions.

2.5 Implementation Issues

Ideally, for portability reasons, one we would like to implement all system components using only Java. This approach turns out, however, to be currently infeasible for several reasons. The most important are security constraints imposed on Java. Using pure Java model we would have to create a client-server model in which all communication and data distribution are located in one host. This idea had to be rejected because of its inflexibility.

Performance and advanced functionality is another constraint. At least for now, high-end 3D visualization or video encoding is impractical in Java. This may be a transient situation. Digital signatures, Java performance improvements, and a growing body of advanced libraries written in Java may allow us to in the future to implement the entire system in this language. For the time being, we felt that the benefit of building a flexible, powerful system with rich functionality outweighs aesthetic architectural concerns.

Daemon Implementation

We use the LiveConnect technology introduced by Netscape Inc. in its Navigator version 3.0. This product provides means for Netscape components such as Java applets, JavaScript scripts and plug-ins to talk to each other. The *daemon* has been implemented as a plug-in. Using LiveConnect mechanisms, each applet residing in the same page with the plug-in has access to the plug-in handle. Message passing between plug-in and the applet is achieved by calling

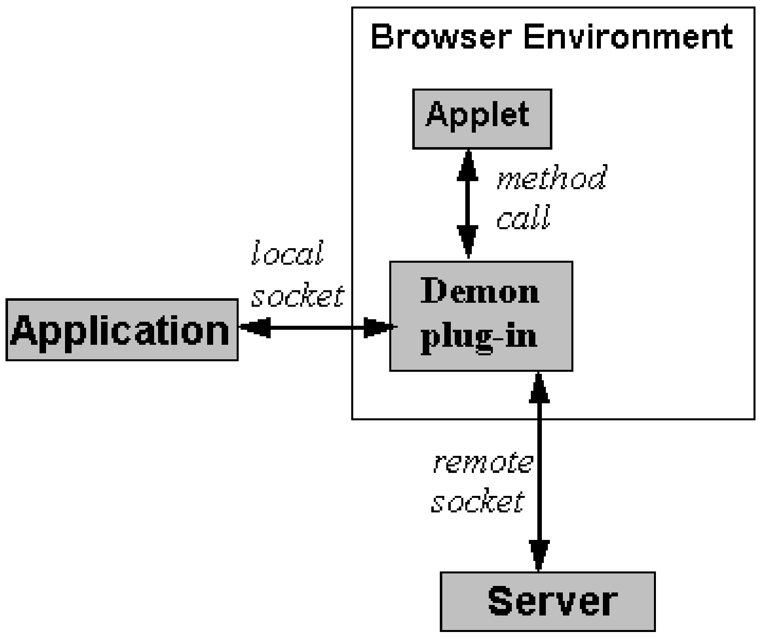


Figure 4: TANGO Communication Scheme

appropriate methods of each other. In case of standalone applications, the connection to the daemon is implemented with use of standard socket communication (Figure 4).

Plug-ins are usually written in C (the API provided by Netscape uses this language). To improve the porting procedure of this—the only platform dependent component of TANGO we associate Java code with the native C code and implement large parts of the plug-in code in Java. The only the part of the plug-in implemented is C is the set of routines used to start standalone applications and to provide intranode applet-application message passing. In this way we have minimized porting effort to an absolute minimum. TANGO is currently available on both Unix and Wintel platforms.

Application API

Porting applications to TANGO is facilitated by a simple API [TANGO:97a], and libraries currently available for C, C++, and Java. A detailed API description is beyond the scope of this article. The basic functionality includes `register`, `write`, and `receive` as well as a set of functions used to control state of the applets. In case of Java we provide a package containing two classes: the `AppletBase` class implements functions mentioned above and the `Message` class provides a wrapper for user-defined messages. For C language the API is defined as a set of functions provided in a library. The message itself is implemented as a structure. For a complete description of TANGO API, consult [TANGO:97a].

At present, messages are represented by byte arrays. It is the programmer's responsibility to convert these data properly. Object serialization and RMI (Remote Method Invocation) will be used as soon as these technologies are available from JavaSoft (Java 1.1 capability).

Server Implementation

The *central server* is a multithreaded standalone Java application. The server keeps all the information about connections and users in dynamic data structures and also stores them in the database. In our implementation we use JDBC (Java Data Base Connection) package and Oracle RDBMS.

Videoteleconferencing

For scalability reasons, the real time multimedia streams are not sent via central server. Instead, a distributed architecture akin to the Insoft's (currently Netscape) OpenDVE has been implemented [Stachowiak:97a] . This architecture supports multicast. Session control remains with the TANGO session manager. Audio/video decode/playback capability is implemented in Java for certain supported codecs. Backend database repository and playback capability for the audio/video streams is provided by a random access video server and is a part of the global system session archiving/retrieval facility.

3 Application of TANGOsim to Command and Control

Command and Control is the military description of a general real time decision (or judgment) support environment involving a complex set of people, datasets, and computational resources. A critical characteristic is the need to make the “best possible,” as opposed to “optimal,” choice. In a civilian context, crisis management [NRC:96a] has essentially similar needs. However, most application areas have a component that links computers, information, and one or more people to make decisions. Correspondingly, command and control has system requirements that are generally applicable.

We are implementing a simple scenario corresponding to an unmanned aircraft flying into U.S. airspace carrying a bacterial agent. We implement planning, execute (does one intercept or not?), and finally response phases to health and life threatening consequences of the downed aircraft. This scenario involves military and civilian radar sensors and personnel tracking aircraft; NORAD as military command in execution stage; federal and state leaders at highest level (President and Governor); weather simulation to assess intercept possibilities, and in response stage, the dispersal of bacterial agents; FEMA as civilian command in response stage; and finally, medical authorities for expertise and treatment.

Let us see how some of the key capabilities of TANGOsim are used in those various parts of this scenario. Generally, digital video conferencing, text-based chatboards, and shared white boards are used by the participants to interact in a typical unstructured collaborative fashion.

One of the simplest shared applications is the common information source that, in our case, is a shared Web page, but in general, can be information obtained from any database. This is, for instance, used by the tracking officer to search the Web for information on the identified aircraft, and display characteristics and photos with other participants. Both the Web search process, and resultant pages are shared. Note this information is generated by one *master*, and broadcast to other participants. We would use the filter capability of TANGOsim to ensure that each participant only received appropriate information. For instance, the President and Governor would be spared a lot of the detailed shared displays used by those lower down the command chain.

More generally, one wishes to present a given object in different ways to different participants. Thus, the radar and weather officers could use complex three-dimensional geographical information systems to study the event in detail. The expected weather and tracking data would be presented to others less involved in those areas, as simpler two-dimensional summaries. This flexibility is easily implemented by passing all status messages from the master for each application through a filter that is controlled by a dynamic script. Note that one will need to select different masters for different applications, and allow the "master role" be passed from one participant to another as the collaboration evolves. This capability is supported by TANGOsim.

Training is an important component of many decision making processes, including command and control, crisis management, and health care. Here, the discrete event simulation capability of TANGOsim is critical as it allows one to "script" (simulate) any of the participants, and enable general training sessions focusing on any role. Note that in the simple TANGO collaborative mode shown in Figure 1(a), "time" is the real wall clock time. This simple mode will support some scripting, which can be used to present information at particular times to given clients. For instance, this is a valuable capability in the training mode for carousel-like presentations. In the more general simulation mode, one differentiates real and virtual time as is traditional in discrete event simulators.

Command and control, and crisis management both make decisions about events in the real world, and so particularly critical applications are two- and three-dimensional geographical information (GIS) systems. We believe that Web technology is excellent for GIS, and had started development of these for education—the virtual field trip. We originally implemented our three-dimensional GIS in VRML (Virtual Reality Modelling Language), but found this unacceptedly slow. We re-implemented the VRML version in terms of its Open Inventor basis, which led to a much faster C++ application. This we can still integrate into TANGOsim which supports any client application language.

As usual in GIS, these applications support overlays—in the case of the three-dimensional GIS, these include weather simulations with volume rendering of the clouds, and other weather phenomena. We are currently extending this to other three-dimensional fields, including electromagnetic simulations.

4 Other Applications of TANGOsim

In the previous section, we highlighted some capabilities of TANGOsim and described their use in our current command and control project. Here, we will more speculatively discuss the role of this technology in health care, computational steering, and distance education.

Health care has, like command and control, interesting training applications. Today, the latter could involve a mix of say residents in training, experts, observers, and a set of test cases that form the shared information space. Clearly, TANGOsim would allow a distributed implementation of this with the experts playing the role of "anchor desks" in the crisis management notation. An important characteristic of this, and other applications is the number of participants as the current TANGOsim involves a single centralized Java server. This is not a scalable model, and future systems will surely involve a Web of cooperating servers. Turning to team health care, we see the need for both spontaneous collaboration between members of the initial care team, and later asynchronous collaboration as specialists are called in for later consultations. A key feature of TANGOsim is the ability to log and replay collaborative sessions. This is obviously of great potential value (and some legal concern) in health care and other applications. A key capability that must be added is a notation so that initial sessions are abstracted by the initial participants for the later experts. We have some experience with this in education where lengthy audio clips are available for elaboration of summary presentations. Of course, TANGOsim supports traditional asynchronous collaboration, including multimedia mail.

In computational steering, TANGOsim is particularly suitable for team activities, such as grand challenge simulations or access to a remote instrument for a group of investigators. The scripted feature of TANGOsim could link to a resource allocation system, such as CODINE, LSF, NQS, DQS or Condor (http://nhse.cs.rice.edu/NHSEreview/96-1.html), which would activate client applications when the resource was available. In industrial applications, such as vehicle design, many engineers may be involved in a multidisciplinary optimization with different roles. This illustrates how one can use server filters to present each engineer only those components of the linked computation that are of relevance to them. Inherent in TANGOsim is the use of Web technologies for visualization. As we have discussed (http://www.npac.syr.edu/projects/javaforcse), we believe Java visualization and data analysis "wrappers" will prove to be an attractive way for users to interact with conventional or high performance simulation servers. We have also shown how one can emulate and generalize popular visualization systems, such as AVS or Khoros with the WebFlow concept, which can be naturally integrated with TANGOsim, as both are based on Java servers. Finally, we note that the current NSF PACI program involves collaborative support for distributed computation. This is, again, an area where we expect Web technologies to be very important.

Finally, we mention educational applications. Many sites have demonstrated the great value of the Web in disseminating information—Cornell's Virtual Workshop is a good example (http://www.tc.cornell.edu/Edu/VW). However, education involves both curricula material and a rich complex interaction between teachers and students. This involves both synchronous (deliver lectures) and asynchronous (grade homework, answer questions) collaboration. We see that systems like TANGOsim provide a Web framework for addressing the collaborative component of education (the missing link?). It is clear that we can provide for more attractive features than traditional video based distance education, but it will require many experiments to identify which capabilities of Web collaborative systems are needed with what functionalities. These developments could be very important for the education community because it is possible that successful deployment of Web collaboration systems could alter the basic (business) model for education. One critical point is scaling—how many simultaneous students can one and does one want to support in a single session. Reliability and performance will also be critical—it is essential that server interruptions will be very rare and that one minimizes transmission delays. Good caching of the basic information so that the network is only used in real time for dynamic interaction seems essential here. We are currently investigating some of these issues with some rather modest experiments. It is important that many different such trials be undertaken for although the general concept is clear, it is not either clear how to implement the details, and how important are either the general or particular capabilities.

References

[Beca:97a] Beca, Lukasz, Cheng, Gang, Fox, Geoffrey C. Jurga, Tomasz, Olszewski, Konrad, Podgorny, Marek, Sokolowski, Piotr, Stachowiak, Tomasz, and Walczak, Krzystof. “TANGO—A Collaborative environment for the World-Wide Web.” Submitted to *Proceeedings of the Fifth High Performance Distributed Computing*, August, 1997. Syracuse University Technical Report.

[Chandy:97a] Chandy, K. M., and Rifkin, A. “Systematic composition of objects in distributed internet applications,” in *Proceedings of the 30th Hawaii International Conference on System Sciences*, January 1997. Maui, Hawaii.

[CIP:96] Caltech Infospheres Project (1996) http://www.infospheres.caltech.edu/

[Gosling:89a] Gosling, J., Rosenthal, D. S. H., and Arden, M. (1989) *The NeWS Book*, Springer-Verlag.

[Gosling:96] Gosling, B. Joy, and G. Steele. (1996) The Java Language Specification. Addison-Wesley Developers Press, Sunsoft Java Series

[Grudin:90] Grudin, J. (1990) Computer-supported cooperative work: history and focus. IEEE Computer 27: 19 - 26

[NCSA:96] NCSA Habanero Project (1996)
http://www.ncsa.uiuc.edu/SDG/Software/Habanero/

[NRC:96a] Steering Committee. “Computing and communications in the extreme: Research for crisis management and other applications,” in *Workshop Series on High Performance Computing and Communications.* National Academy Press, Washington, D.C., 1996. Computer Science and Telecommunications Board, Commission on Physical Sciences, Mathematics, and Applications.

[Schooler:96] Schooler E.M. (1996) Conferencing and Collaborative Computing. Multimedia Systems 4: 210-225

[Stachowiak:97a] Stachowiak, T. (1997), “Development of ??? Web-based Index Teleconferencing System,” *NPAC Conferencing System*, January 1997.
http://trurl.npac.syr.edu/toms/Projects/NCS/ncs-info.html

[TANGO:96a] TANGO Collaboratory (1996). http://www.npac.syr.edu/tango/

[TANGO:97a] Tango API for Developers (1997).
http://www.npac.syr.edu/tango/TangoAPI.html

[Warner:95a] Warner, D., Sale, J., and Viirre, E. “A systems approach for developing an integrative healthcare information distribution infrastructure.” Technical Report SCCS-749, Syracuse University, NPAC, Syracuse, NY, November 1995.