CEWES MSRC/PET TR/99-15

Using WebHLA to Integrate HPC FMS Modules with Web/Commodity based Distributed Object Technologies of CORBA, Java, COM and XML

by

Geoffrey C. Fox
Wojtek Furmanski
Ganesh Krishnamurthy
Hasan T. Ozdemir
Zeynep Odcikin Ozdemir
Tom A. Pulikal
Krishnan Rangarajan
Ankur Sood

Work funded by the DoD High Performance Computing Modernization Program CEWES Major Shared Resource Center through

Programming Environment and Training (PET)

Supported by Contract Number: DAHC94-96-C0002
Nichols Research Corporation

Using WebHLA to Integrate HPC FMS Modules with Web/Commodity based Distributed Object Technologies of CORBA, Java, COM and XML

Geoffrey C. Fox, Ph. D., Wojtek Furmanski, Ph. D.,

Ganesh Krishnamurthy, Hasan T. Ozdemir, Zeynep Odcikin-Ozdemir, Tom A. Pulikal, Krishnan Rangarajan, Ankur Sood

Northeast Parallel Architectures Center, Syracuse University
111 College Place, Syracuse University, Syracuse NY 13244-4100
{gcf, furm, gkrishna, timucin, zeynep, tapulika, krrangar, asood} @ npac.syr.edu

Keywords:
Interactive Simulation, Military, Personal Computers, Standards

ABSTRACT

HLA standards for interoperability between various DoD Modeling and Simulation paradigms are being enforced in parallel with the rapid onset of new Object Web / Commodity standards for distributed objects and componentware, emergent at the crossroads of CORBA, COM, Java, and XML technologies. WebHLA explores synergies between and integrates both trends by offering Object Web based implementation of the HLA framework. Our goal is to deliver a uniform platform that facilitates conversion of legacy codes to and development of new codes in compliance with HLA, HPC and Object Web standards. We outline here the overall design of WebHLA, we summarize the system components prototyped so far and we illustrate our approach for one HPC FMS application – Parallel CMS (Comprehensive Mine Simulator) - in the area of large scale minefield simulation and countermine engineering.

1. INTRODUCTION

We present here early results of our work on Web / Commodity based High Performance Modeling and Simulation, conducted as part of the academic branch of the Forces Modeling and Simulation (FMS) domain within the DoD HPC Modernization Program. Our approach explores synergies between ongoing and rapid technology evolution processes such as: a) transition of the DoD M&S standards from DIS and ALSP to HLA; b) extension of Web technologies from passive information dissemination to interactive distributed object computing offered by CORBA, Java, COM and W3C WOM; and c) transition of HPCC systems from custom (such as dedicated MPPs) to commodity base (such as NT clusters).

One common aspect of all these trends is the enforcement of reusability and shareability of products or components based on new technology standards. DMSO HLA makes the first major step in this direction by offering the interoperability framework between a broad spectrum of simulation paradigms, including both real-time and logical time models (DMSO 1998).

However, HLA standard specification leaves several implementation decisions open and to be made by the application developers - this enables reusability and integrability of existing codes but often leaves developers of new simulations without enough guidance.

In WebHLA, we fill this gap by using the emergent standards of Web based distributed computing – we call it *Pragmatic Object Web* (Orfali 1998; Fox et al. 1999) - that integrate Java, CORBA, COM and W3C WOM models for distributed componentware as illustrated in Fig. 1.

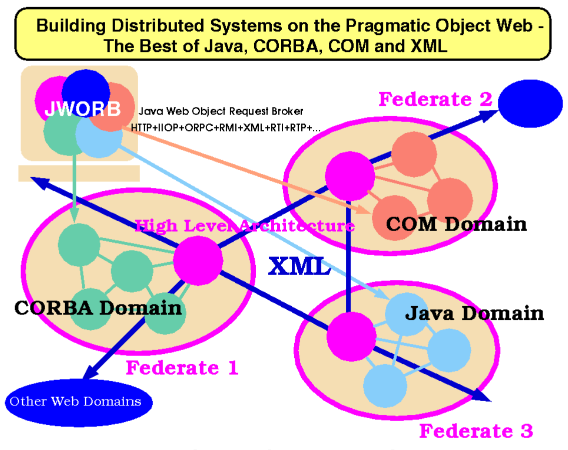

Building Distributed Systems on the Pragmatic Object Web - The Best of Java, CORBA, COM and XML
JWORB
Java Web Object Request Broker
HTTP+IIOP+ORPC+RMI+XML+RTI+RTP+...
Federate 2
High Level Architecture
COM Domain
XML
CORBA Domain
Java Domain
Federate 1
Other Web Domains
Federate 3

Figure 1: *Pragmatic Object Web concepts and components.*

We believe that WebHLA, defined as the convergence point of the standardization processes outlined above will offer a powerful modeling and simulation framework, capable to address the new challenges of DoD computing in the areas of Simulation Based Design, Testing, Evaluation and Acquisition.

In this document, we summarize the WebHLA architecture (Section 2), we review the status of WebHLA components as of Feb '99 (Section 3) and we illustrate our approach on example of one large scale HPC FMS application – Parallel CMS (Section 4).

2. WEBHLA OVERVIEW

The overall architecture of WebHLA follows the 3-tier architecture of our *Pragmatic Object Web* (Fox et al. 1999) (see Figure 1) with a mesh of JWORB (Java Web Object Request Broker) based middleware servers, managing backend simulation modules and offering Web portal style interactive multi-user front-ends. JWORB is a multi-protocol server capable to manage objects conforming to various distributed object models and including CORBA, Java, COM and XML. HLA is also supported via *Object Web RTI* (OWRTI) i.e. Java CORBA based implementation of DMSO RTI 1.3, packaged as a JWORB service. As illustrated in Fig. 1, objects in any of the popular commodity models can be naturally grouped within the WebHLA framework as HLA federates and they can naturally communicate by exchanging (via JWORB based RTI) XML-ized events or messages packaged as some suitable FOM interactions.

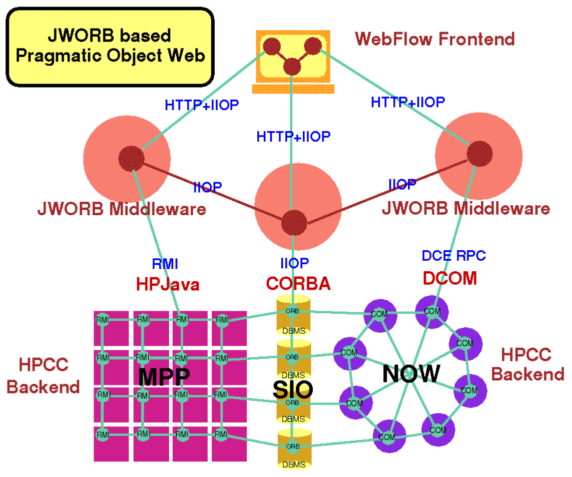


Figure 2: *Example of protocol integration support in JWORB.*

HLA-compliant M&S systems can be integrated in WebHLA by porting legacy codes (typically written in C/C++) to suitable HPC platforms, wrapping such codes as WebHLA federates using cross-language (Java/C++) RTICup API, and using them as plug-and-play components on the JWORB/OWRTI software bus. In case of previous generation simulations following the DIS (or ALSP) model, suitable bridges to the HLA/RTI communication domain are also available in WebHLA, packaged as utility federates. To facilitate experiments with CPU-intense HPC simulation modules, suitable database tools are available such as event logger, event database manager and event playback federate that allow us to save the entire simulation segments and replay later for some analysis, demo or review purposes. Finally, we also constructed SimVis – a commodity (DirectX on NT) graphics based battlefield visualizer federate that offers real-time interactive 3D front-end for typical DIS=>HLA entity level (e.g. ModSAF style) simulations.

In the following, we describe in more detail in Chapter 3 the WebHLA components listed above, followed (in Chapter 4) by an example of using the system to integrate a realistic large scale HPC FMS simulation.

3. WebHLA COMPONENTS

3.1 JWORB based Object Web RTI

DMSO's longer range plan includes transferring HLA to industry as CORBA Facility for Modeling and Simulation.

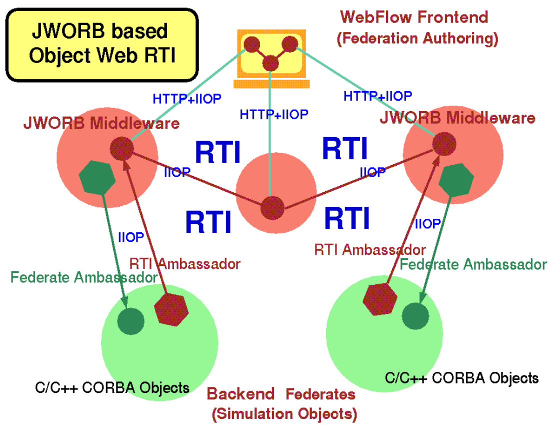


Figure 3: *Top view representation of the Object Web RTI.*

Anticipating these developments, we have recently developed in one of our HPCMP FMS PET projects at NPAC an Object Web based RTI (Fox et al.1998c) prototype, which builds on top of our new JWORB (Java Web Object Request Broker) middleware integration technology. JWORB is a multi-protocol Java network server, currently integrating HTTP (Web) and IIOP (CORBA) and hence acting both as a Web server and a CORBA broker (see Fig. 2) Such server architecture enforces software economy and allows us to efficiently prototype new interactive Web standards such as XML, DOM or RDF in terms of an elegant programming model of Java, while being able to wrap and integrate multi-language legacy software within the solid software engineering framework of CORBA.

We are now testing this concept and extending JWORB functionality by building Java CORBA based RTI implementation structured as a JWORB service and referred to as *Object Web RTI* (see Fig. 3). Our implementation includes two base user-level distributed objects: RTI Ambassador and Federate Ambassador, built on top of a set of system-level objects such as RTI Kernel, Federation Execution or Event Queues (including both time-stamp- and receive-order models). RTI Ambassador is further decomposed into a set of management objects, maintained by the Federation Execution object, and including: Object Management, Declaration Management, Ownership Management, Time Management and Data Distribution Management.

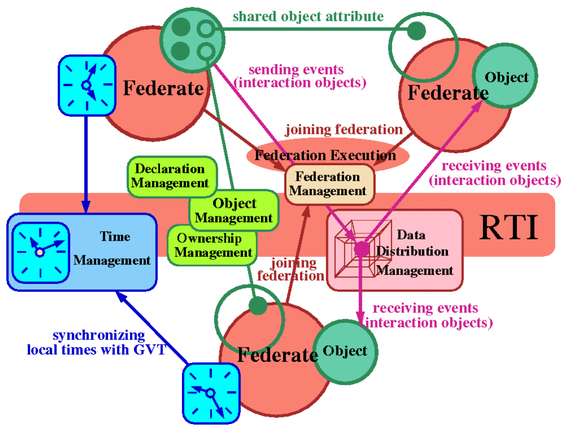


Figure 4: *RTI services (rounded rectangles) implemented as CORBA objects in Object Web RTI.*

RTI is given by some 150 communication and/or utility calls, packaged as 6 main management services: Federation Management, Object Management, Declaration Management, Ownership Management, Time Management, Data Distribution Management, and one general purpose utility service. Our design is based on 9 CORBA interfaces, including 6 Managers, 2 Ambassadors and RTIKernel. Since each Manager is mapped to an independent CORBA object (see Fig. 4), we can easily provide minimal support for distributed management by simply placing individual managers on different hosts.

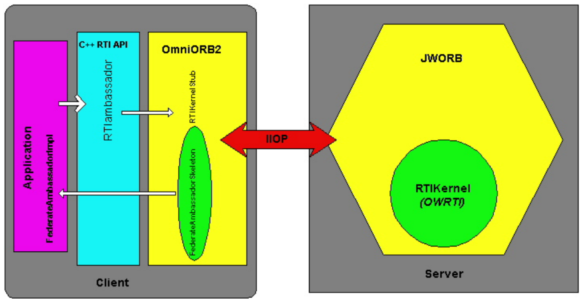


Figure 5: *Architecture of C++ RTI API that allows to link C++ RTI clients with Java RTI services of OWRTI.*

To be able to link C++ clients with OWRTI, we developed a C++ library (see Fig. 5) which: a) provides RTI C++ programming interface; and b) it is packaged as a CORBA C++ service and, as such, it can easily cross the language boundaries to access Java CORBA objects that comprise our Java RTI. Our C++ DMSO/CORBA glue library uses public domain OmniORB2.5 as a C++ Object Request Broker to connect RTI Kernel object running in Java based ORB. RTI Ambassador glue/proxy object forwards all method calls to its CORBA peer and Federate Ambassador, defined as another CORBA object running on the client side, forwards all received callbacks to its C++ peer. This library is running on Windows NT, IRIX and SunOS systems.

3.2 Parallel ports of selected M&S modules

In parallel with prototyping core WebHLA technologies described above, we are also analyzing some selected advanced M&S modules such as the CMS (Comprehensive Mine Simulator) system developed by Steve Bishop's team at Ft. Belvoir, VA that simulates mines, mine fields, minefield components, standalone detection systems and countermine systems including ASTAMIDS, SMB and MMCM. The system can be viewed as a virtual T&E tool to facilitate R&D in the area of new countermine systems and detection technologies of relevance both for the Army and the Navy. We recently constructed a parallel port of the system to Origin2000, where it was packaged and can be used as either a DIS node or as an HLA federate.

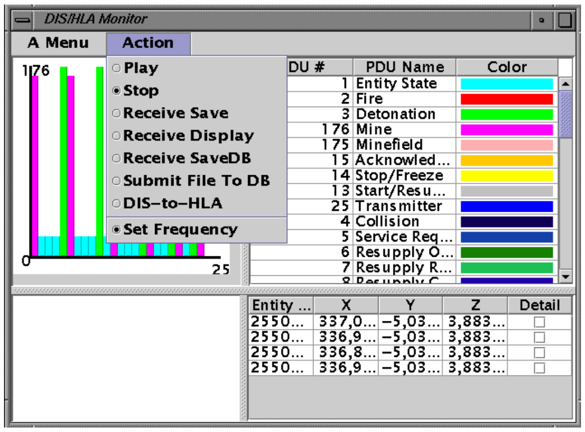
Origin systems support a variety of parallel programming tools. These include: a) Parallel Compilers that take sequential Fortran, C or C++ codes, optionally with some user-provided compiler directives (pragmas), and they return the corresponding parallel codes, generated in either fully automatic and semi-automatic (compiler directives based) parallelization modes; b) Message Passing libraries such as MPI, PVM, and Cray SHMEM; c) Scientific & Math Libraries as made available in the Silicon Graphics Cray Scientific Library (SCSL) and other third party libraries; d) Operating system-based inter-process and inter-thread communication via standards-based sockets, pthreads, and shared memory.

Based on the analysis of the sequential CMS code, we found the semi-automatic, compiler directives based approach as the most practical parallelization technique to start with in our case. The most CPU-intensive inner loop of the CMS simulation runs over all mines in the system and it is activated in response to each new entity state PDU to check if there is a match between the current vehicle and mine coordinates that could lead to a mine detonation. Using directives such as 'pragma parallel' and 'pragma pfor' we managed to partition the mine-vehicle tracking workload over the available processors, and we achieved a linear speedup up to four processors. For large multiprocessor configurations, the efficiency of our pragmas based parallelization scheme deteriorates due to the NUMA memory model. Indeed, on a distributed shared memory architecture such as Origin, the latency for a CPU to access main memory increases with the distance to the physical memory accessed and with the contention on the internal network. In consequence, the cache behavior of the program has a significant impact on the performance. To assure scalability across the whole processor range, we need therefore to enforce data decomposition that matches the already accomplished workload/loop decomposition. Our initial experiments with enforcing such full decomposition by using a combination of pragma directives did not succeed, most likely due to rather complex object oriented and irregular code processed in the CMS inner loop. We are currently rebuilding and simplifying the inner loop code so that the associated memory layout of objects is more regular and hence predictable. For example we are replacing linked lists over dynamic object pointers by regular arrays of statically allocated objects etc.

Having ported the CMS code to Origin2000, we also converted it from DIS to HLA simply by mapping all PDUs to the corresponding HLA interactions and by using C++ RTI API that offers connectivity between the C++ and Java RTI models. An HLA port abstraction layer was introduced that allows for easy switch between DIS and HLA communication modes with the minimal modification of the original code. CMS Federate Ambassador receives interactions from RTI and it passes them to the HLA port that translates them into DIS PDU, to be further parsed internally by the legacy CMS code. In a similar way, HLA port translates PDUs generated internally by the CMS into HLA interactions before passing them to the RTI Ambassador.

3.3 JDIS: DIS-HLA Bridge and Event I/O Manager

DIS/HLA bridge for CMS described above was constructed internally inside the CMS federate code. In an alternative approach, explored for ModSAF modules, we use another DIS-HLA bridge that operates as an independent process and acts as DIS node on the input channel and as an HLA federate on the output channel. We constructed such a bridge called JDIS in Java, starting from the free DIS Java parser offered by the dis-java-vrml working group of the Web3D Consortium and completing it to support all PDUs required by the linked ModSAF + CMS simulators.



DU #	PDU Name	Color
1	Entity State	🟦
2	Fire	🟥
3	Detonation	🟩
176	Mine	🟪
175	Minefield	🟥
15	Acknowled...	🟧
14	Stop/Freeze	🟨
13	Start/Resu...	⬜
25	Transmitter	🟦
4	Collision	🟦
5	Service Req...	🟦
6	Resupply O...	🟩
7	Resupply R...	🟩
8	Resupply C...	🟦

Entity ...	X	Y	Z	Detail
2550...	337,0...	−5,03...	3,883...	☐
2550...	336,9...	−5,03...	3,883...	☐
2550...	336,8...	−5,03...	3,883...	☐
2550...	336,9...	−5,03...	3,883...	☐

Figure 6: *JDIS Front-End control and display panel*.

JDIS provides linkage between DIS applications such as ModSAF and CMS running as HLA application. JDIS receives DIS PDUs produced by ModSAF, translates them into HLA interactions and lets RTI forward them to CMS.

JDIS can also write / read PDUs from a file and hence it can be used to log and playback sequences of simulation events. We also used this tool to generate point-like PDU probes as well as stress test PDU sequences when testing and measuring performance of Parallel CMS.

Visual front-end of JDIS, illustrated in Fig. 6, supports runtime display of the PDU flow, and it offers several controls and utilities, including: a) switches between DIS, HLA and various I/O (file, database) modes; b) frequency calibration for a PDU stream generated from file or database; c) PDU probe and sequence generators; d) simple analysis tools such as statistical filters or performance benchamrks that can be performed on accumulated PDU sequences.

In order to facilitate the transmission of PDUs and their storage in the Database, we adopted XML as a uniform wire format and specified a process for converting all PDUs to XML format. More specifically, we defined an interface called *XML-izabl*e and a class *Protocol Data Unit* that implements this interface. This interface defines methods for reading, writing, inserting and extracting PDUs in an XML format in the Document Object Model.

We have written implementations for converting most of the PDUs in the XML format. These include the Entity State, Detonation, Collision, MineField, Transmitter, Receiver, Acknowledge, Designator, Fire, Repair Response, Repair Complete, Resupply Cancel, Resupply Received, Signal, Service Request, Signal, Start / Resume, Stop / Freeze, Mine, MineField Request, MineField Response, MineField State and MineFeld Nack.

There is an ongoing standardization effort within the HLA community called *A Real-Time Platform Reference FOM* (RPR-FOM) with the aim to define a FOM that offers complete mapping of DIS. Our JDIS will fully comply with RPR-FOM after the standard specification if completed.

3.4 PDUDB: Event DB Logger and Playback Federate

Playing the real scenario over and over again for testing and analysis is a time consuming and tedious effort. A database of the equivalent PDU stream would be a good solution for selectively playing back segments of a once recorded scenario. For a prototype version of such a PDU database we used Microsoft's Access database and Java servlets for loading as well as retrieving the data from the database using JDBC.

The PDU logger servlet receives its input via HTTP PORT message in the form of XML-encoded PDU sequences. Such input stream is decoded, converted to SQL and stored in the database using JDBC. The DIS header field common to all the PDUs is stored in a separate table from the PDU bodies. This table has all the attributes in the DIS header like the PDU type, timestamp, etc. When the PDUs are generated from the database this timestamp value is used for calculating the frequency with which the PDUs are to be send. Some PDUs have some variable number of attributes like articulation parameters etc., and these attributes are stored in separate tables so the entire database is normalized.

The Playback is done using another servlet that sends the PDUs generated from the database as a result of a query. The servlet is activated by accessing it from a web browser. Currently the queries are made on timestamps. But any possible queries can be made on the database to retrieve any information. The servlet can send the PDUs either in DIS mode or in HLA mode.

PDU-DB also has a dynamic HTML interface to the tables and data stored in the database. The interface lets you select any of the PDU tables and browse quickly and efficiently through the individual records or groups in the table.

3.5 SimVis: DirectX based Battlefield Visualizer

In our Pragmatic Object Web approach, we integrate CORBA, Java, COM and WOM based distributed object technologies. We view CORBA and Java as most adequate for the middleware and backend, whereas COM as the leading candidate for interactive front-ends due to the Microsoft dominance on the desktop market.

Of particular interest for the M&S community seems to be the COM package called DirectX which offers multimedia API for developing powerful graphics, sound and network play applications, based on a consistent interface to devices across different hardware platforms.

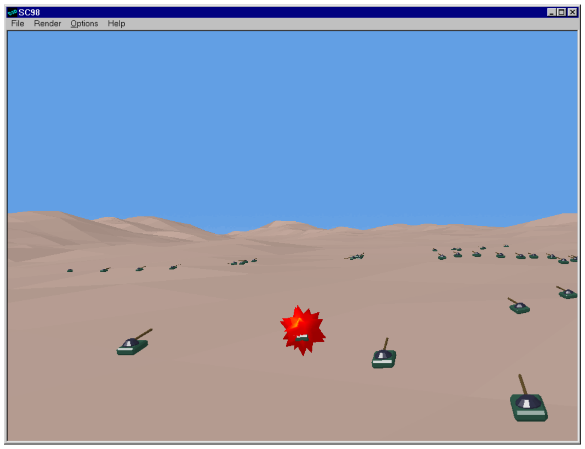


Figure 7: *Sample screen from CMS simulation in SimVis.*

Using DirectX/Direct3X technology, we constructed a real-time battlefield visualizer, SimVis (see Fig. 7) that can operate both in the DIS and HLA modes. SimVis an NT application written in Visual C++ and it contains the following components: a) internal HLA-DIS bridge, constructed in a similar way as for Parallel CMS discussed above; b) a fast Winsock library based PDU parser; c) Direct3D based rendering engine. PDU parser extracts the battlefield information from the event stream, including state (e.g. velocity) of vehicles in the terrain, position and state of mines and minefields, explosions that occur e.g. when vehicles move over and activate mines etc. The parser performs also suitable type conversions for the network to NT data formats, it constructs the suitable memory data structures and it passed them to the viewer.

The viewer was created using the DirectX/Direct3D API. The ModSAF terrain is the input for a sampler program, which provides vertices for each of the faces. The sampler program also provides the colors and texture type for each face of the terrain. Using these data, faces are constructed & added to the terrain mesh. Then normals are generated for these faces and the terrain is constructed. After construction, the terrain is added as a visual object to the scenario scene. Geometry objects and animation sets for typical battlefield entities such as armored vehicles (tanks) and visual events such as explosions were developed using 3D Studio MAX authoring system.

SimVis visual interactive controls include base navigation support in terms of directional keys (left, right, up, down, Home, End) as well as some other modes and options, including: a) various rendering modes (wireframe, flat, Gouraud; b) mounting the camera/viewport on a selected vehicle or plane; c) several scene views such as Front, Top, Right, Left and Satellite Views.

4. APPLICATION EXAMPLE: PARALLEL CMS

We illustrate now how all WebHLA components described above cooperate in one specific HPC FMS application – Parallel CMS.

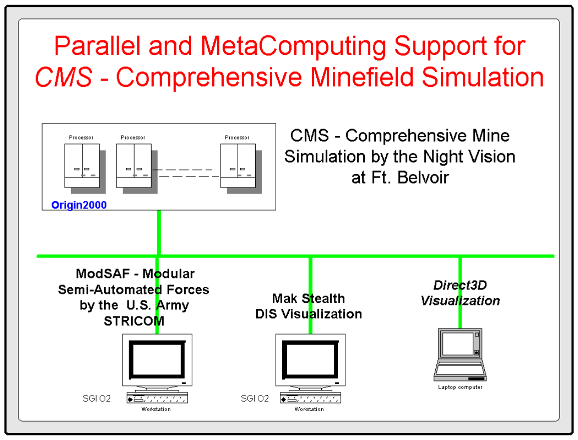

Parallel and MetaComputing Support for
CMS - Comprehensive Minefield Simulation
Processor
Processor
Processor
Origin2000
CMS - Comprehensive Mine Simulation by the Night Vision at Ft. Belvoir
ModSAF - Modular Semi-Automated Forces by the U.S. Army STRICOM
Mak Stealth DIS Visualization
Direct3D Visualization
SGI O2
Workstation
SGI O2
Workstation
Laptop computer

Figure 8: *Parallel CMS demo at SC'98 in Orlando, FL.*

CMS is an advanced DIS system under development by the Night Vision Lab at Ft. Belvoir, VA. CMS simulates a broad spectrum of mines and minefield to interact with vehicles such as those provided by ModSAF, on the virtual battlefield. Modern warfare can require millions of mines to be present on the battlefield, such as in the Korean Demilitarized Zone or the Gulf War. The simulation of such battlefield arenas requires High Performance Computing support. Syracuse University is building Parallel and Metacomputing Support for CMS by porting the CMS module to Origin2000 and linking it with a collection of distributed simulators handling terrain, vehicles and visualization.

Our early results for Parallel CMS were demonstrated at Supercomputing '98. The overall configuration of the demonstrated system, still based on the DIS communication and the multicast/MBONE networking is illustrated in Fig. 8 and it includes: a) Parallel CMS running on Origin2000; b) a set of ModSAF vehicles running on SGI workstation; and c) real-time 3D visualization front-ends, including Mak Stealth and our SimVis tool described before.

More recently, we completed the process of converting Parallel CMS from a DIS node to an HLA federate, and we also constructed the JDIS bridge that allowed us to effectively treat ModSAF nodes as HLA federates. Finally, we also completed the PDUDB federate that allows us to log simulation events (DIS PDUs or the equivalent HLA interactions) in an SQL database and reply the whole simulation or its selected/filtered segments on demand.

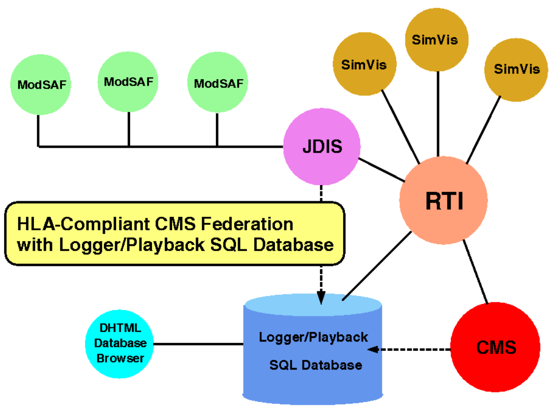

ModSAF
ModSAF
ModSAF
SimVis
SimVis
SimVis
JDIS
RTI
HLA-Compliant CMS Federation with Logger/Playback SQL Database
DHTML Database Browser
Logger/Playback SQL Database
CMS

Figure 9: *Architecture of WebHLA based Parallel CMS.*

The overall configuration of the most recent,. HLA-compliant version of Parallel CMS system is presented in Fig 9. Parallel CMS federate runs on Origin2000 at ARL MSRC, other modules (ModSAF, JDIS, PDUDB, SimVis) are running on NPAC SGI and NT workstations. Using JDIS, we can easily switch between DIS and HLA modes and between real-time and playback simulation modes. The latter is useful for analysis and demonstrations. In particular, we also constructed a mobile laptop demo in the playback mode with Microsoft Access based PDUBD federate, Javasoft JDK based JDIS, and DirectX based SimVis.

5. SUMMARY

We presented here our approach towards building WebHLA as an interoperable simulation environment that implements new DoD standards on top of new Web / Commodity standards. In particular, we illustrated how HLA/RTI can cooperate with Java (used in JWORB, OWRTI, JDIS), CORBA (used in JWORB, OWRTI), XML (used as alternative representation for FED files and as a wire format for DIS PDUs) and Mocrosoft COM (used via DirectX in SimVis front-end). Parallel CMS described above is a rather sophisticated FMS application and yet we were able to construct it in a low budget academic R&D environment by making the optimal use and exploring fully the synergy between all involved Web/Commodity technologies listed above.

Hence, our initial results are encouraging and we therefore believe that WebHLA with evolve towards a powerful modeling and simulation framework, capable to address new challenges of DoD and commodity computing in many areas that require federation of multiple resources and collaborative Web based access such as Simulation Based Design and Acquisition.

6. ACKNOWLEDGEMENTS

This work was partially funded by the DoD High Performance Computing Modernization Program's Programming Environment and Training (PET) project, including ARL Major Shared Resource Center support, Contract Number: DAHC 94-96-C-0010 with Raytheon Systems Company, and CEWES Major Shared Resource Center support, Contract Number: DAHC 94-96-C-0002 with Nichols Research Corp.

7. REFERENCES

Defense Modeling and Simulation Office (DMSO) 1998, High Level Architecture, http://hla.dmso.mil.

D. Bhatia, V. Burzevski, M. Camuseva, G. Fox, W. Furmanski and G. Premchandran, 1997, WebFlow - a visual programming paradigm for Web/Java based coarse grain distributed computing , Concurrency: Practice and Experience, vol. 9, no. 6, June 97, 555-577.

Robert Orfali and Dan Harkey, 1998, Client/Server Programming with Java and CORBA , 2nd Edition, Wiley.

G. C. Fox, W. Furmanski, H. T. Ozdemir and S. Pallickara, 1999, Building Distributed Systems for the Pragmatic Object Web, Wiley, (in progress).

G. C. Fox, W. Furmanski, S. Nair, H. T. Ozdemir, Z. Odcikin Ozdemir and T. A. Pulikal, WebHLA - An Interactive Programming and Training Environment for High Performance Modeling and Simulation, In Proceedings of the SISO Simulation Interoperability Workshop, SIW Fall 98, paper SIW-98F-216, Orlando, FL, Sept 14-18, 1998)

G. C. Fox, W. Furmanski and H. T. Ozdemir, 1998, Java/CORBA based Real-Time Infrastructure to Integrate Event-Driven Simulations, Collaboration and Distributed Object / Componentware Computing, In Proceedings of PDPTA'98 (Las Vegas, NV, July 98.

G. C. Fox, W. Furmanski, S. Nair, H. T. Ozdemir, Z. Odcikin Ozdemir and T. A. Pulikal, WebHLA - An Interactive Multiplayer Environment for High Performance Distributed Modeling and Simulation, In Proceedings of the International Comference on Web-based Modeling and Simulation, WebSim99, San Francisco, CA January 17-20 1999.