Java/CORBA based Real-Time Infrastructure to Integrate Event-Driven Simulations, Collaboration and Distributed Object/Componentware Computing

G.C. Fox, W. Furmanski, and H.T. Ozdemir
Northeast Parallel Architectures Center, Syracuse University, Syracuse NY, USA

Abstract
We are discussing the four major standard candidates for distributed object/componentware computing: Java, CORBA, COM and WOM within our proposed coordination framework we call Pragmatic Object Web (POW). We describe our integration approach based on multi-protocol middleware server JWORB (Java Web Object Request Broker) that currently integrates HTTP and IIOP and which we now further develop to also support COM and WOM core functionalities. We are also experimenting with visual dataflow authoring front-ends using NPAC WebFlow system on top of JWORB based software bus. Finally, we illustrate our technologies in one major application domain - DoD Modeling and Simulation - where we use JWORB to implement the Real-Time Infrastructure (RTI) layer of High Level Architecture (HLA). HLA was recently specified by DMSO as a general integration framework for DoD distributed simulations and we claim that we can bring it to a broader community of distributed collaborative object/componentware computing via the interactive Web/CORBA/Java/COM interfaces of our Pragmatic Object Web.

Keywords: CORBA, Java, COM, WOM, Web Object Management Architecture, Pragmatic Object Web, Visual Dataflow, WebFlow, Modeling and Simulation, High Level Architecture, Run-Time Infrastructure.

1 Pragmatic Object Web

Recent developments in Internet/Intranet technologies start influencing the whole field of distributed computing, both in its enterprise and science & engineering domains. Most notably, Java appeared during the last few years as the leading language candidate for distributed systems engineering due to its elegant integrated support for networking, multithreading and portable graphical user interfaces.

While the "Java Platform" or "100% Pure Java" philosophy is being advocated by Sun Microsystems, industry consortium led by the OMG pursues a multi-language approach built around the CORBA model. It has been recently observed that Java and CORBA technologies form a perfect match as two complementary enabling technologies for distributed system engineering. In such a hybrid approach, referred to as Object Web [1], CORBA is offering the base language-independent model for distributed objects and Java offers a language-specific implementation engine for the CORBA brokers and servers.

Meanwhile, other total solution candidates for distributed objects/components are emerging such as DCOM by Microsoft or WOM (Web Object Model) by the World-Wide Web Consortium. However, standards in this area and interoperability patterns between various approaches are still in the early formation stage. For example, recent OMG/DARPA workshop on compositional software architectures [2] illustrated the growing momentum, the multitude of options, and at the same time the uncertainty of the overall direction in the field. A closer inspection of the distributed object/component standard candidates indicates that, while each of the approaches claims to offer the complete solution, each of them in fact excels only in specific selected aspects of the required master framework. Indeed, it seems that WOM is the easiest, DCOM the fastest, pure Java the most elegant and CORBA the most realistic complete solution.

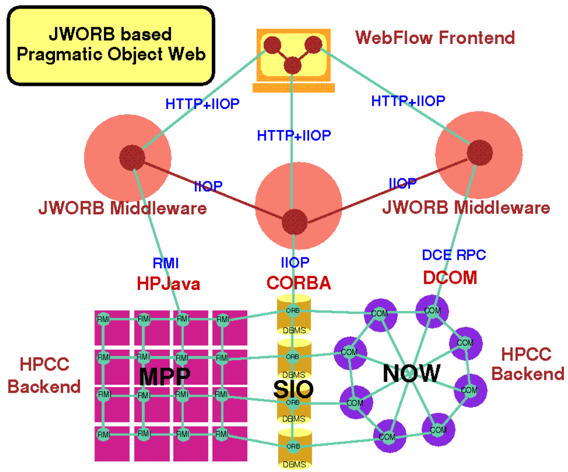

Figure 1: *Support for multiple protocols and heretogeneous backend services within the JWORB based Pragmatic Object Web.*

In our Pragmatic Object Web [3] approach at NPAC we adopt the integrative methodology i.e. we setup a multiple-standards based framework in which the best assets of various approaches accumulate and cooperate rather than competing. We start the design from the middleware which offers a core or a 'bus' of modern 3-tier systems and we adopt Java as the most efficient implementation language for the complex control required by the multi-server middleware. We adopt CORBA as the base distributed object model at the Intranet level, and the (evolving) Web as the world-wide distributed (object) model. System scalability requires fuzzy, transparent boundaries between Intranet and Internet domains which therefore translates into the request of integrating the CORBA and Web technologies. We implement it by building a Java server (JWORB) [4] which handles multiple network protocols and includes support both for HTTP and IIOP. On top of such Pragmatic Object Web software bus, we implement specific computational and collaboratory services.

2 JWORB (Java Web Object Request Broker) based middleware

JWORB [4] is a multi-protocol extensible server written in Java. The base server has HTTP and IIOP protocol support. It can serve documents as an HTTP Server and it handles the IIOP connections as an Object Request Broker. As an HTTP server, JWORB supports base Web page services, Servlet (Java Servlet API) and CGI 1.1 mechanisms. In its CORBA capacity, JWORB is currently offering the base remote method invocation services via CDR based IIOP and we are now implementing the Interface Repository, Portable Object Adapter and selected Common Object Services.

After the core JWORB server starts up, it looks at configuration file to find out which protocols are supported and it loads the necessary protocol classes for each protocol (Definition, Tester, Mediator, Configuration). Definition Interface provides the necessary Tester, Configuration and Mediator objects. Tester object looks at the current connection's stream and decides whether it can interpret this connection or not. Configuration object is responsible for the configuration parameters of a particular protocol. Mediator object serves the connection. New protocols can be added simply by implementing the four classes described above and by registering a new protocol with the JWORB server.

After JWORB accepts a connection, it asks each protocol handler object if it can recognize this protocol. If JWORB finds a handler which claims that it can serve this connection, then this protocol handler deals with this connection. Current algorithm looks at each protocol according to their order in the configuration file. This process can be optimized with randomized or prediction based algorithm. At present, HTTP and IIOP messaging are supported and the current protocol is simply detected based on the magic anchor string value (GIOP for IIOP and POST, GET, HEAD etc. for HTTP).

3 WOMA (Web Object Management Architecture) based Integration

As discussed above, our multi-standard integration framework starts from the Java based CORBA middleware and then gradually incorporates other emergent standard candidates, using the OMA services and facilities as the implementation base. The advantage of this approach is that new dynamic features of the emergent models for distributed objects/components can be quickly and reliably prototyped using Java in the solid software engineering framework of CORBA.

For example, we are building at NPAC a visual dataflow authoring environment - WebFlow [5] - for composing Web based distributed applications from reusable modules. Early WebFlow prototype was based on custom Java API for the module interface. New JWORB based WebFlow under development attempts at standards based visual componentware support and it addresses the integration of the CORBA component, DCOM component and the Enterprise JavaBeans models. Armed with a Java based CORBA platform such as JWORB to be soon augmented by the CORBA/COM bridge, we will be able to freely experiment with and mix-and-match all these component standard candidates.

We are also analysing several other emergent standard candidates coming from the Web, Enterprise, Desktop of Military domains and we are exploring their integration patterns within the new WebFlow framework. We discuss some of these activities below.

In the visual graph editing sector, UML appeared recently as a new promising standard candidate, adopted by OMG and also embraced by Microsoft. We therefore intend to base new WebFlow authoring model on the

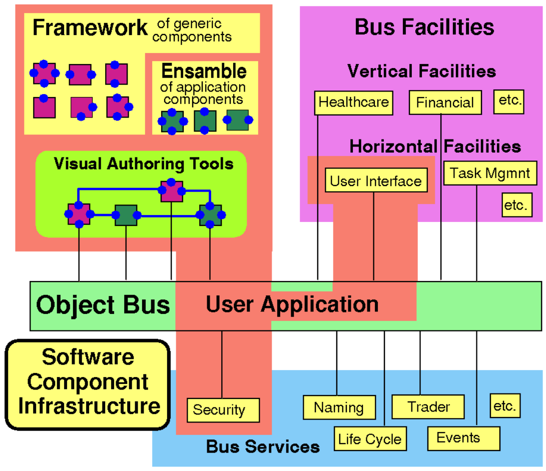

Figure 2: *Visual dataflow authoring tools as a high-level front-end for a software bus based distributed computing middleware (*for example WebFlow over JWORB as discussed in Section 4*)*

UML model, extended towards runtime support for UML activity and collaboration diagrams.

The WebFlow operating environment, represented in the current prototype by custom Session-, Module- and Connection Managers, shares in fact several common features with the standard RTI layer of the DMSO HLA discussed in the next Section. It seems therefore reasonable to build the new WebFlow runtime around the JWORB based RTI communication dynamics, thereby assuring close interoperability with the new generation interactive systems such as on-line gaming.

W3C introduces its own distributed object/componentware framework, sometimes referred as WOM, and emergent as a combination of XML, RDF and DOM standards. We are currently evaluating XML as a possible candidate for the universal data format for inter-modular transmission channels in WebFlow. We also intend to incorporate DOM as a CORBA service and to integrate RDF with our efforts towards transparent persistence across UNIX and PC platforms discussed below.

We coined the term WOMA (Web Object Management Architecture) which can be viewed as a merger of W3C WOM and OMG OMA and we use to refer to our multi-standards based integration approach. Basically, rather than debating if we should follow OMA, Java, COM, WOM etc. we are setting instead an evaluation testbed and an integration framework that will let these technologies to work in concert. Such process will clearly expose the synergy as well as the essental differences between various approaches and hopefully it will contribute itself to enforcing and enabling common standards for the Web, Desktop, Enterprise and Defense computing. Some of our current activities, aimed at integrating the alternative standard candidates for the front-end, middleware and the backend layers, include:

- Integrating DirectX/Java3D/VRML to support commodity desktop frontends
- Integrating DCOM with Object Web via COM/CORBA bridges
- Integrating OLEDB/JDBC/PSS to support transparently persistent backends
- Integrating new DoD Standards for Modeling and Simulation - High Level Architecture - with the commodity standards of Pragmatic Object Web.

In this paper, we elaborate on the last activity which we view as potentially valuable and relevant for several other domains of distributed computing and collaboration.

4 High Level Architecture and Run-Time Infrastructure

High Level Architecture (HLA) [6] under development by the Defense Modeling and Simulation Office (DMSO) offers a common integration and interoperability platform for a broad spectrum of simulation paradigms. These include real-time (DIS) models used for combat training simulations, logical-time / event-driven models used for forces simulation, and faster-than-real-time models used in analysis simulations.

HLA is a distributed object technology with the object model defined by the Object Model Template (OMT) specification and including the Federation Object Model (FOM) and the Simulation Object Model (SOM) components. HLA FOM objects interact by exchanging HLA interaction objects via the common Run-Time Infrastructure (RTI) acting as a software bus similar to CORBA. Current HLA/RTI follows a custom object specification but DMSO's longer term plans include transferring HLA to industry via OMG CORBA Facility for Interactive Modeling and Simulation.

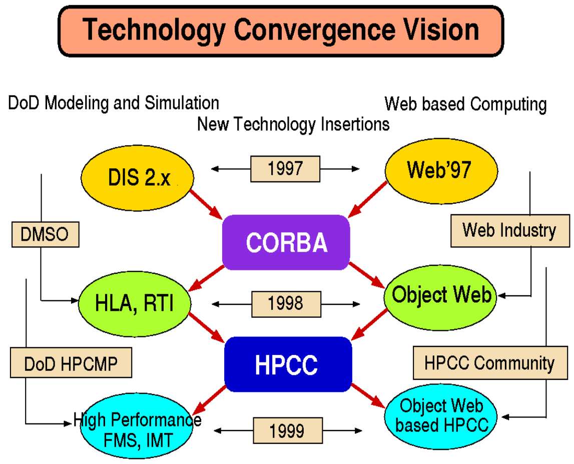

Figure 3: *A roadmap illustrating synergies between technology evolution processes within the DoD Modeling and Simulation, Object Web Technologies and High Performance Computing.*

At NPAC, we are anticipating these developments are we are building a prototype RTI implementation in terms of Java/CORBA objects using the JWORB middleware.

Although coming from the DoD computing domain, RTI follows generic design patterns and is applicable to a much broader range of distributed applications, including modeling and simulation but also collaboration, on-line gaming or visual authoring. From the HPCC perspective, RTI can be viewed as a high level object based extension of the low level messaging libraries such as PVM or MPI. Since it supports shared objects management and publish/subscribe based multicast channels, RTI can also be viewed as an advanced collaboratory framework, capable of handling both the multi-user and the multi-agent/multi-module distributed systems [7][8].

In the following, we summarize the ongoing NPAC work on JWORB based RTI prototype implementation.

5 JWORB based RTI Prototype at NPAC

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, we can easily provide minimal support for distributed management by simply placing individual managers on different hosts.

The communication between simulation objects and the RTI bus is done through the RTIambassador interface. The communication between RTI bus and the simulation objects is done by their FederateAmbassador interfaces. Simulation developer writes/extends FederateAmbassador objects and uses RTIambassador object obtained from the RTI bus.

RTIKernel object knows handles of all manager objects and it creates RTIambassador object upon the federate request. Simulation obtains the RTIambassador object from the RTIKernel and from now on all interactions with the RTI bus are handled through the RTIambassador object. RTI bus calls back (asynchronously) the FederateAmbassador object provided by the simulation and the federate receives this way the interactions/attribute updates coming from the RTI bus.

Federation Manager object is responsible for the life cycle of the Federation Execution. Each execution creates a different FederationExecutive and this object keeps track of all federates that joined this Federation.

Object Manager is responsible for creating and registering objects/interactions related to simulation. Federates register the simulated object instances with the Object Manager. Whenever a new registration/destroy occurs, the corresponding event is broadcast to all federates in this federation execution.

Declaration Manager is responsible for the subscribe/publish services for each object and its attributes. For each object class, a special object class record is defined which keeps track of all the instances of this class created by federates in this federation execution. This object also keeps a seperate broadcasting queue for each attribute of the target object so that each federate can selectively subscribe, publish and update suitable subsets of the object attributes.

Each attribute is currently owned by only one federate who is authorized for updating this attribute value. All such value changes are reflected via RTI in all other federates. Ownership Management offers services for transferring,maintaining and querying the attribute ownership information.

We are currently working on implementing the Time Management service which offers support for logical time handling, and the data Distribution Management which offers advanced publish/subscribe services via routing spaces or multi-dimensional regions in the attribute value space.

In parallel with the first pass prototype implementation, we are also addressing the issues of more organized software engineering in terms of Common CORBA Services. We are now testing this concept and extending JWORB functionality by building Java

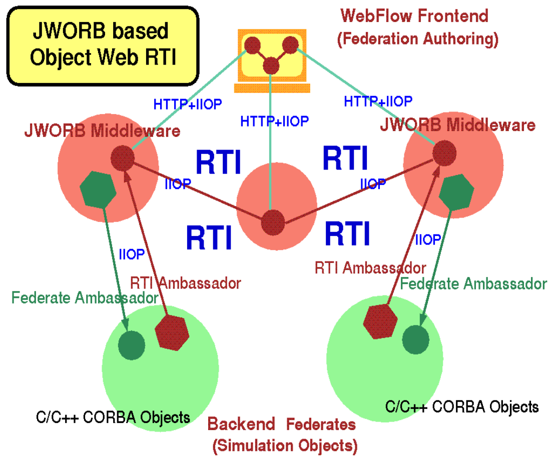

Figure 4: *Object Web RTI constructed as a set of CORBA objects (RTIAmbassador at the server side, FederateAmbassador at the client side) managed by the JWORB middleware and editable by visual dataflow authoring tools in the front-end.*

CORBA based RTI implementation structured as a JWORB service and referred to as Object Web RTI [4]. 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. To be able to run C++ RTI demo examples, we developed a C++ library 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.

For example, we intend to use the CORBA Naming Service to provide uniform mapping between the HLA object names and handles, and we plan to use CORBA Event and Notification Services to support all RTI broadcast/multicast mechanisms. This approach will assure quality of service, scalability and fault-tolerance in the RTI domain by simply inheriting and reusing these features, already present in the CORBA model.

6 Outlook

We outlined here our Pragmatic Object Web based integration approach and we discussed in more detail the NPAC implementation of the RTI software bus in the JWORB framework. We intend to use our RTI both for DoD-specific high performance M&S applications within our WebHLA [9] project with the DoD High Performance Computing Modernization Office, and for other interactive distributed Web/Commodity application domains such as synchronous and asynchronous collaboration at visual dataflow authoring environments.

More generally, we believe that our JWORB+WOMA integration approach will offer an efficient testbed for evaluating and possibly integrating new emergent Web/Commodity technologies. Given the ongoing competition between the major standards bodies, such a testbed operated within an academic lab linked with several federal programs such as NPAC might play a role itself in the formation process of Web/Commodity standards for distributed object/componentware computing.

References

[1] 1. Robert Orfali and Dan Harkey, Client/Server Programming with Java

and CORBA , 2nd Edition, Wiley 1998.

[2] Craig Thompson, OMG/DARPA Workshop on Compositional Software Architectures, Monterey, CA January 6-8 1998

[3] G. C. Fox, W. Furmanski and S. Pallickara, Building Distributed Systems for the Pragmatic Object Web, book in progress, Wiley '98.

[4] G. C. Fox, W. Furmanski and H. T. Ozdemir, JWORB - Java Web Object Request Broker for Commodity Software based Visual Dataflow Metacomputing Programming Environment , submitted for the HPDC-7, Chicago, IL, July 28-31, 1998.

[5] D. Bhatia, V. Burzevski, M. Camuseva, G. Fox, W. Furmanski and G. Premchandran, WebFlow - a visual programming paradigm for Web/Java based coarse grain distributed computing , June '97, in the special issue of "Concurrency: Practice and Experience" on Java for Scientific Computing.

[6] High Level Architecture (HLA) by the Defence Modelling and Simulation Office (DMSO)

[7] D. Dias, G. C. Fox, W. Furmanski, V. Mehra, B. Natarajan, H. T. Ozdemir, S. Pallickara, Z. Ozdemir, Exploring JSDA, CORBA and HLA based MuTech's for Scalable Televirtual (TVR) Environments , presented at the Workshop on OO and VRML, VRML98 Conference, Monterey, CA, Feb 16-19,1998.

[8] G.C. Fox, W. Furmanski, B. Natarajan, H. T. Ozdemir, Z. Odcikin Ozdemir, S. Pallickara and T. Pulikal, Integrating Web, Desktop, Enterprise and Military Simulation Technologies To Enable World-Wide Scalable Televirtual (TVR) Environments , submitted to the Workshop on Web-based Infrastructures for Collaborative Enterprises, the WET ICE'98 Conference, Stanford University, June 17-19,1998.

[9] D. Bernholdt, G. C. Fox, W. Furmanski, B. Natarajan, H. T. Ozdemir, Z. Odcikin Ozdemir and T. Pulikal, WebHLA - An Interactive Programming and Training Environment for High Performance Modeling and Simulation , submitted to the DoD HPC 98 Users Group Conference, Rice University, Houston, TX, June 1-5 1998.