Using Web Services to Integrate Heterogeneous Simulations in a Grid Environment

J. Mark Pullen[1], Ryan Brunton[2], Don Brutzman[3], David Drake[2], Michael Hieb[4], Katherine L. Morse[2], and Andreas Tolk[5]

[1] Department of Computer Science & C3I Center, George Mason University, Fairfax, Virginia, 22030, USA
mpullen@gmu.edu
[2] Science Applications International Corporation (SAIC), 10260 Campus Point Drive, San Diego, CA 92121, USA
{RYAN.P.BRUNTON,DAVID.L.DRAKE-2,KATHERINE.L.MORSE}@saic.com
[3] MOVES Institute, Naval Postgraduate School, Monterey, California 93943 , USA
brutzman@nps.navy.mil
[4] Alion Science & Technology, S &T, 1901 N. Beauregard St. Alexandria, Virginia 22311, USA
Michael.R.Hieb@us.army.mil
[5] VMASC, Old Dominion University, Norfolk, Virginia 23529, USA
atolk@odu.edu

Abstract. The distributed information technologies collectively known as Web services recently have demonstrated powerful capabilities for scalable interoperation of heterogeneous software across a wide variety of networked platforms. This approach supports a rapid integration cycle and shows promise for ultimately supporting automatic composability of services using discovery via registries. This paper presents a rationale for extending Web services to distributed simulation environments, including the High Level Architecture (HLA), together with a description and examples of the integration methodology used to develop significant prototype implementations. A logical next step is combining the power of Grid computing with Web services to facilitate rapid integration in a demanding computation and database access environment. This combination, which has been called Grid services, is an emerging research area with challenging problems to be faced in bringing Web services and Grid computing together effectively.

1 Introduction

The distributed information technologies collectively known as Web services recently have been shown to offer powerful capabilities for scalable interoperation of heterogeneous software across a wide variety of networked platforms. A particular appeal of this approach is the rapid integration cycle it provides, which shows promise for supporting automatic composability of services ultimately by using discovery on Web service registries. The authors have championed extension of Web services to distributed simulation environments, including the HLA, under the name Extensible Modeling and Simulation Framework (XMSF)[1]. We present below our

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3038, pp. 835–847, 2004.

rationale for the viability of the approach and a description of the integration methodology we have used to develop significant prototype implementations. XMSF has the potential to become a unifying framework for heterogeneous distributed simulation involving a wide variety of languages, operating systems, and hardware platforms. Interoperation of non-HLA simulations with HLA federations and other software systems such as military command and control is an attractive possibility.

We are now engaged in expanding the XMSF paradigm to address a wider range of simulation issues. An overarching approach to system design using the Model Driven Architecture (MDA) in the context of web service implementations is proving to be a foundational standard, as presented in Section 3. Connecting high-performance computational resources to diverse physics-based 3D visualizations is increasingly important. An example is shown in Section 4.3 where Sonar is visualized by computation and rendering of sonar-ping propagation via a Web service. XMSF has the potential to become a unifier for heterogeneous distributed simulation involving a wide variety of languages, operating systems, and hardware platforms. As the scale and complexity of such systems increases, the required computational, database, and networking resources soon will outstrip the collections of networked workstations we have used to create our prototypes. A logical extension is to bring together Web services with Grid computing, as addressed in the final section of this paper.

2 Web Services Definition

The fundamental idea behind Web services is to integrate software applications as services. This concept is based on a defined set of technologies, supported by open industry standards, that work together to facilitate interoperability among heterogeneous systems, whether within an organization or across the Internet. In other words, Web services enable applications to communicate with other applications using open standards. This approach has tremendous potential to build bridges between "stove-piped" legacy systems, that is, systems that were designed without open lateral interfaces to each other. Fundamentally, Web services provide an approach to distributed computing with application resources provided over networks using standard technologies. Because they are based on standard interfaces, Web services can communicate even if they are running on different operating systems and are written in different languages. The standards are widely supported by industry and have been applied successfully in a wide range of different domains. For this reason, Web services provide an excellent approach for building distributed applications that must incorporate diverse systems over a network.

The Web services framework provides a set of operations, modular and independent applications that can be published, discovered, and invoked by using industry standard protocols described below. The resulting distributed computing model represents the interaction between pairs of programs, rather than the interaction between programs and user. An equivalent statement is that Web services are discrete Web-based applications that interact dynamically with other Web services. In order to make this happen, several sub-functions are necessary:

Self-description of the service functionality
Publishing the service descriptions using a standardized format

Locating the service with the required functionality
Establishing message communications with the service
Requesting the required data to initiate the service
Exchanging data with other Web services, including delivering the results.

Protocols most commonly used for these purposes are the Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL), and Universal Distribution Discovery and Interoperability (UDDI).

2.1 Messaging for Web Services

All web service implementations consist of three core components: a consumer, a provider, and an optional service registry. The consumer application locates providers either by querying a registry service or by accessing a known service endpoint. Service requests are encoded using an XML vocabulary understood by both the consumer and the provider, encapsulated in a SOAP envelope, and sent to the provider using Internet protocols. The message is decoded, acted on, and the response encoded and returned in a parallel process. While the consumer/provider abstraction is an effective descriptive simplification, the roles are not always so straightforward. SOAP envelope headers can contain routing information that allows messages to be propagated in a chain of connected web services, each node modifying the payload and acting as both a provider for the previous node and a consumer for the succeeding node. Alternatively, in a system consisting of two bidirectional nodes like the Web Enabled RTI, each node is both a provider and a consumer. Figure 1 illustrates these relationships.

While the most common implementation of Web services sends messages in SOAP over the Hypertext Transfer Protocol (HTTP) used for webpages, the formal W3C definition is far more robust and flexible:

> A Web service is a software system identified by a URI [RFC 2396], whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.[2].

Given this definition, Web services bindings are not constrained by the traditional limitations of HTTP based Web services. For example, as a request-response based protocol, the HTTP is inherently limited to operations that take place synchronously within a brief connection time. Protocols such as Simple Mail Transfer Protocol (SMTP) and the Blocks Extensible Exchange Protocol (BEEP) allow for asynchronous message processing and response. We will consider here the most general form of Web service.

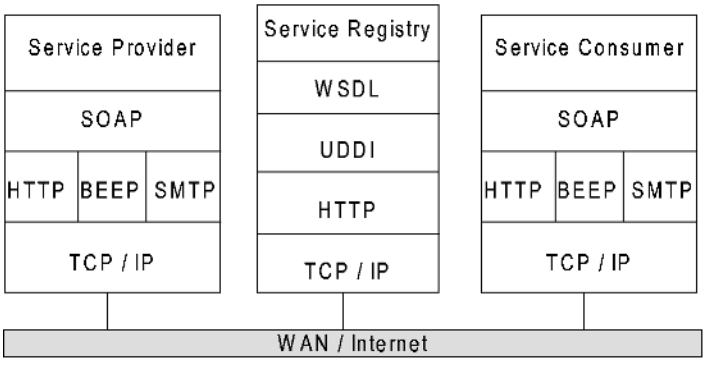

Fig. 1. Web Services Protocol Stack

2.2 Web Services and the HLA-RTI

The High Level Architecture (HLA) for simulation, as defined in IEEE Standard 1516 [3], provides a structure and set of services for networked interoperation of simulation software. The interoperation strategy is based on defining a Federation Object Model (FOM) that is common to all members of a simulation federation and providing a Run Time Infrastructure (RTI) that supports a standard set of communication services among the federates. We have proposed framework (XMSF) that expands the concepts of the HLA into the realm of Web services. In the most general case, we envision each simulation providing one or more Web services that provide the data needed to interoperate with other simulations. The rules of the HLA restrict federates to sharing FOM data via the federation's RTI, but do not preclude sharing data with other software or other federations by means of a Web service offered through the RTI.

Moreover, interoperability of software via Web services depends on creating a common vocabulary (in effect, an ontology) that is shared by two or more software applications and expressed as an XML tagset. This arrangement can be seen as a superset of the FOM, which also can be expressed as an XML tagset under IEEE standard 1516. Thus we see that the stage is already set for adapting Web services to support HLA-compliant distributed simulations. Benefits might occur in two ways:

- The RTI supporting an HLA federation can be constructed from commercial Web service components, reducing cost to support the HLA because the cost of commercial components is amortized over a larger number of users.
- HLA federations can interoperate with other software applications supported by Web services, using as a tagset a mapping between the federation's FOM and whatever ontology is shared with the other applications. This approach offers considerable promise for interoperation of military command, control, communications, computing and intelligence (C4I) software whose sponsors have not been willing to adopt the HLA but are beginning to use via Web services as an interoperability solution.

3 Composing Simulation Systems Using the MDA Techniques

For the XMSF approach to be viable, we must show how it can bridge the gap between HLA-compliant federations and other software systems and also the gap that arises when simulations are supported by different underlying infrastructures, such as traditional systems and the Grid computing approach. To achieve this will require a combination of a technical approach and an overarching conceptual approach, allowing composition and orchestration of the participating systems. While Web services are sufficient on the technical level, Web services alone are insufficient to achieve this at a conceptual level; therefore XMSF advocates applying techniques from the Model Driven Architecture of the Object Management Group (OMG) [4] to facilitate design of meaningful interoperability among distributed and individually implemented components.. Applying the MDA approach can provide benefits in structuring functionality in a cross-platform, cross-language way using well-defined patterns for model based system design and re-engineering. Conscientious use of well-defined design patterns, as exemplified by MDA principles, provides a basis on which predictably interoperable and composable software components can be assembled. This holds true both within complex applications and across interconnected applications running across the Internet.

The Web service vision is that services will work together seamlessly because they are developed to the same standards for self-description, publication, location, communication, invocation, and data exchange capabilities. As all the standards concerned are open, the technologies chosen for Web services are inherently neutral to compatibility issues that exist between programming languages, middleware solutions, and operating platforms. As a result, applications using Web services can dynamically locate and use necessary functionality, whether available locally or from across the Internet.

However, the existence of the appropriate standards is not sufficient to ensure consistency and meaningful interoperability in the distributed simulation environment. As currently defined, the Web services family of standards does not support the composition of agile and dynamic simulation components. Meaningful implementation of interoperability among simulation systems on this level requires composability of the underlying conceptual models. The HLA and related approaches provide the mechanics of interoperation, but they fall short when aiming at composability [5][6]. Conceptual problems cannot be solved with only implementation-driven solutions, whether they take the form of the HLA or of Web services. To succeed, Web services for distributed heterogeneous simulation must be embedded into a larger context ensuring “meaningful” interoperability. Applying the techniques for software design and re-engineering within a distributed software project can help to ensure composability and orchestration on the conceptual level [4].

Previously the OMG has developed the Common Object Request Broker Architecture (CORBA) standard and related technologies. The MDA approach encompasses the established standards and standard development procedures of OMG and raises them to a new level. The MDA methods ensure that components can be described in a common way, and that the processes of composing these components as well as orchestrating them in a common composition are commonly understood and consistent. The main objective of the MDA is the ability to derive software from a stable model as the underlying infrastructure shifts over time. To achieve this, the

model of the application is captured in an implementation and platform-independent language. The specification of this core model is based on the established OMG standards Unified Modeling Language (UML), Meta-Object Facility (MOF), and the Common Warehouse Metamodel (CWM). The result is a Platform Independent Model (PIM) that captures the concepts of the application rather than its implementation. These PIMs are essential to ensure "meaningful" interoperability, as they capture the concepts and intents of the applications while the other elements the technical connectivity between the components on the communication level. When applied for system design,the PIM is transformed into a Platform Specific Model (PSM) that can be used for CORBA applications, HLA compliant applications, or Web service applications following standardized mapping rules. When applied in the context of re-engineering, following similar inverse mapping rules a PIM can be established as a formal representation of the conceptual model of the component. This PIM can be integrated easily into the formal description of the mission space comprising all PIMs, and hence the concepts and functionality, of all participating components. Therefore, the MDA not only complements the Web services above the implementation level by introducing a common language for conceptual modeling with the PIM, but also supports the migration of legacy components into web enabled components reusable for distributed heterogeneous simulation in future applications [7].

In summary, XMSF is a Web service oriented implementation of the conceptual ideas captured by the MDA. This approach allows legacy systems implemented in heterogeneous information technology environments to migrate toward a common simulation framework, implementing a common conceptual schema within a heterogeneous infrastructure. Thus XMSF not only has the potential to bridge the gap between HLA federations and other systems, but also can facilitate bridging gaps among various hardware and operating system solutions,. We will make the case below that this facilitates Grid applications as well as traditional applications.

4 Implementing Web Services for Simulation

Implementing XMSF requires a shared ontology for data exchange, which takes the form of an XML tagset. As described below, the process of defining this ontology varies considerably according to the organizational environment involved and the complexity of the data to be defined. The process is facilitated considerably by the nature of the Web service model, which involves each software system making its own data available on its own terms. Nevertheless, where the goal is a very broad level of interoperability as in the Command and Control Information Exchange Data Model (C2IEDM) based ontology described below, the process of definition can be quite protracted, as with an HLA FOM. After the tagset has been defined, the strength of the open standards for message-based data exchange becomes evident. We report here on three prototypes, each of which demonstrated rapid, successful integration of existing software.

4.1 XMSF DCEE Viewer

The Distributed Continuous Experimentation Environment (DCEE), managed by the Experimentation Directorate of the U.S. Joint Forces Command (J9/JFCOM), has established a framework of common terminology for the information to be exchanged between components using an enhancement of the Real-time Platform Reference (RPR) FOM. Although the DCEE presently uses HLA as the technical backbone, the concept is open for extensions to emerging solutions. Use of XML, standardized tagsets, Web services, and Internet technology is part of the general concept of DCEE.

Under a project sponsored by JFCOM, we demonstrated the benefits of XMSF in the DCEE with the XMSF DCEE Viewer (XDV). XDV is a web-based, low-cost viewer for DCEE based events in the Joint Semi Automated Forces (JSAF) and relation simulations. The concept for XDV is simple: every eligible stakeholder interested in observing execution of the ongoing experiment can connect to the HLA federation and use the viewer software to follow the actual experiment. The necessary software was installed on widely available computer workstations connected via Internet protocols, allowing eligible stakeholders to follow the experiment from wherever they were located. Key components enabling rapid implementation were the Web Enabled RTI (WE RTI) [8] developed by SAIC and the map graphics packages developed by the Virginia Modeling, Simulation and Analysis Center (VMASC) for visualization as used on command and control maps (unit based viewer) and by the Naval Postgraduate School (NPS) for visualization on the platform level as used in emerging environments (entity based viewer).

Figure 2 illustrates the logical architecture of the XDV. The XDV was tested successfully in all possible four permutations of installation the viewer and server locations at VMASC and JFCOM.

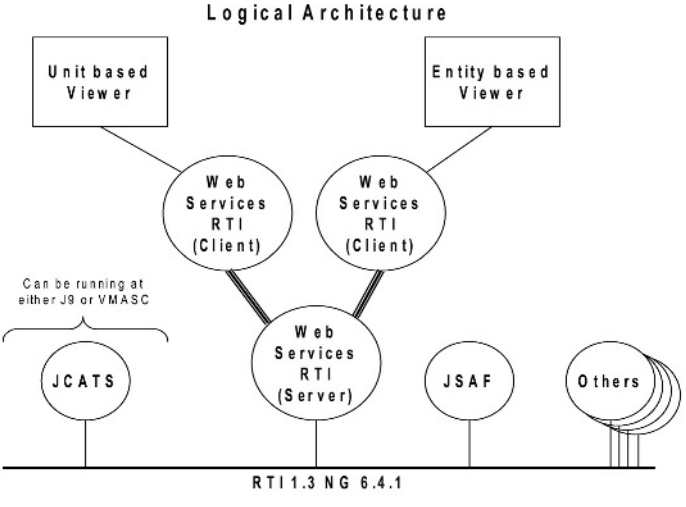

Fig. 2. XDV Logical Architecture

A key future goal is to operationalize the viewer. While the prototype was successful within the limits of its requirements, it is still a prototype, i.e. not robust enough or sufficiently documented to ensure easy distribution, installation, and use by a broad user base. Operationalizing the viewer will entail applying a standard software engineering. We are also investigating the introduction of Area Of Interest Management (AOIM) to support compartmentalization of data and bandwidth use reduction. This work also will be applicable to the Joint National Training Capability (JNTC) currently under development by the Training Directorate of JFCOM.

4.2 Extensible Battle Management Language

Military Command and Control (C2) communication in a network centric environment is postulated to be very data intensive. However, the commander's intent, orders and directives, which is the most critical C2 information, does not currently flow as data. It is communicated as "free text" elements within messages or as stand-alone files. Its vocabulary is found in doctrinal task lists and manuals, but it lacks clearly delineated rules governing its use (semantics and syntax). It is riddled with ambiguity and overlapping definitions. As such, it is incapable of transitioning to the full range of automation that the Department of Defense is implementing. It will not support either digitized C2 or decision support based upon integrated modeling and simulation. While the current formats are suitable for interpersonal communication, they are inadequate for use with simulations, or for future forces that have robotic components. As commanders increasingly rely upon simulation-based decision aids (and therefore use their C2 devices to control simulations) a solution for this "free text" problem must be found. Battle Management Language (BML) was developed as a solution to this problem.

The Battle Management Language (BML) is defined as the unambiguous language used to: 1) command and control forces and equipment conducting military operations and, 2) provide for situational awareness and a shared, common operational picture [9][10]. It can be seen as a representation of a digitized commander's intent to be used for real troops, for simulated troops, and for future robotic forces. The U.S. Army developed a prototype of BML demonstrating the ability to generate an actual National Training Center (NTC) Brigade Operations Order in an advanced Army C4I Planning System, CAPES (Combined Arms Planning and Execution System) and have it executed in widely used Army Simulation, the OneSAF Test Bed (OTB) [11]. The BML prototype used a database containing C4I data, called the Multi Source Data Base (MSDB) to interoperate between the various components in the prototype.

A major achievement of BML was to adapt representations of command and control so that they are directly derived from doctrine and are expressed in a flexible syntax. This provides a means to link the BML (terminology and symbology) directly to their doctrinal source; and it allows operational forces to use their own information systems both to interact with supporting simulations to conduct rigorous, realistic training and support mission rehearsals, and, in the future, to support an expedited military decision making process.

We used the XMSF approach to transform BML to a Web enabled or Extensible Battle Management Language (XBML)[12]. Figure 3 shows Phase One of XBML, which distributed the Army BML Prototype by implementing interfaces to the MSDB

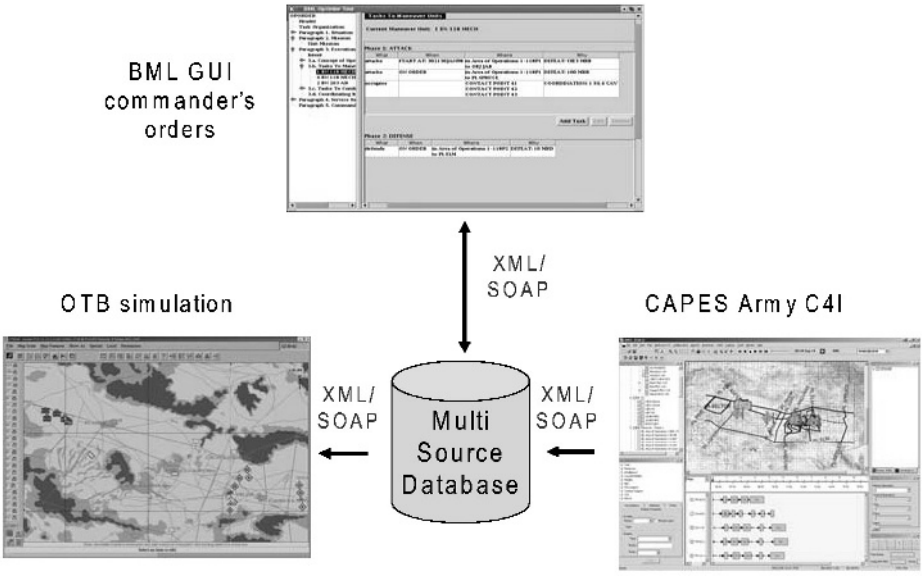

Fig. 3. XBML Testbed project converted to Web services in 3 months

based upon SOAP and XML. This allowed different components to be distributed geographically, and opened the interfaces so more C4I and Simulation nodes could join in an event. The new capability was introduced in less than three months, demonstrating that it is straightforward to rework proprietary interfaces with XMSF standards. Our next goal is to extend XBML into a Joint (involving more than one US military service) and coalition solution, based on open standards. The XML tag set as well as the MSDB of this version is based on the NATO standard, the C2IEDM[13] that will be extended by applying the standards promoted by XMSF. The goal of XBML is to demonstrate a methodology for developing standard doctrinal terms and allowing these to be accessed as Web services.

XBML addresses a longstanding problem in military modeling and simulation: C4I system to simulation interoperability. It does so in two areas:

Development of a Methodology for Easily Integrating Simulations within C4I Systems: The methodology used in "opening" the interfaces in XBML will be documented such that it can be easily reapplied by other members of the community easily, therefore becoming the basis for a standardized way to deal with this challenge.

Defining a Live/C4I Ontology for Simulations: As with other XMSF applications, a common tagset is needed for interoperation. Using the C2IEDM and its standardized procedures for extensions of the model, XBML is contributing to the ontology definitions as the basis of a broadly accepted international standard.

4.3 Sonar Visualization

The NPS Sonar Visualization Project merges physics-based sonar-modeling algorithms, 3D graphics visualization tools and Web-based technologies to provide the military with relevant real-time sonar analysis. Tactical decision aids in this project utilize Web-based Extensible 3D (X3D) models for composable rendering together with Web Services messaging and XML Schema-Based Compression (XSBC) for reliable transmission.

Environmental effects are processed in real-time by distributed sonar servers. Largescale environmental datasets maintained on supercomputers also can be accessed via Web services using XML SOAP messaging.

X3D visualization schemes provide intuitive and useful ways to describe multipath sonar propagation results to USW operators. Animated X3D results are shown in a deployable real-time tactical application. For example, collision detection between sensor raylets and scene geometry aids in evaluation of bottom interactions, surface reflections, and threat contacts.

The project also is developing a common XML tagset to describe acoustic, environmental and sonar path data. This will allow a deployed client to access varying levels of environmental data and processing power, as available, to support specific mission needs.

5 Web Services and Grid

We stated previously that the most common implementation of Web services consists of a SOAP messaging framework using HTTP as the transport mechanism, and that the formal definition of a web service is far more flexible. We now introduce a definition of Grid computing for purposes of comparison:

> A parallel, distributed system composed of heterogeneous resources located in different places and belonging to different administrative domains connected over a network using open standards and protocols [14].

Using this definition, Grid computing can be seen as a natural evolution of distributed computing technologies such as RMI and CORBA (see Figure 4). Where traditional distributed computing technologies tightly coupled the remote resource location to the client stub and often required end-to-end control of the network, Grid computing has moved to a more robust and transparent architecture, allowing resources to discover each other over wide area networks without control of the infrastructure. A lesson also has been learned from the mistakes of those previous proprietary technologies and the positive example provided by Web services. By using open standards and protocols, Grid computing solutions gain the one thing that is necessary for their success: ubiquity of peer resources.

Further comparing the definitions of Grid computing and Web services, we find that while the two both describe distributed computing technologies, they describe them in largely orthogonal ways. The Web services definition focuses on the use of XML to describe both service interfaces and the communication messaging format. Grid computing focuses on the system architecture, leaving the particulars of protocols and message formats unspecified. With complimentary goals and

orthogonal requirements it is unsurprising that Grid computing and Web services have been merged into coherent distributed systems. These solutions, known as Grid services, are Grid computing implementations which use XML to describe their interfaces and encode messages, and open internet protocols for communication. Examples of service implementations that aim to meet this ideal are JXTA, a peer-to-peer architecture supported by Sun Microsystems, and the Open Grid Service Architecture (OGSA), the Web services interface from Globus, whose toolkit is the defacto standard for Grid computing [14][15].

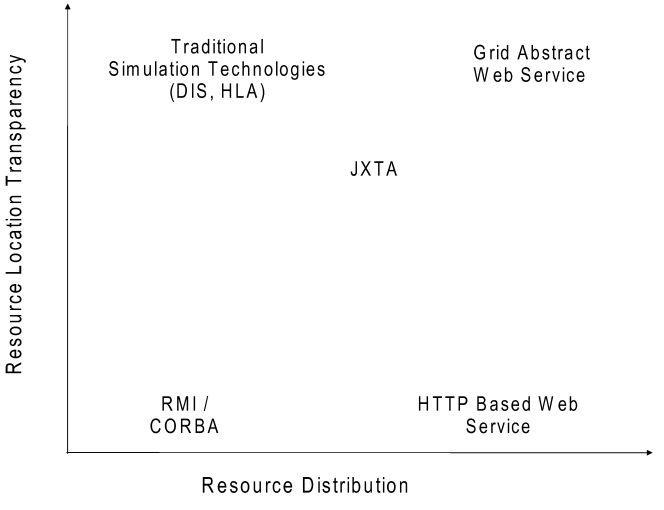

Fig. 4. Comparison of Distributed Computing Paradigms

Having established that Grid computing and distributed HLA based simulations can both be implemented using Web services, we need only discover common ground between Grid computing and the HLA to create a foundation for a Grid services RTI implementation. This common ground exists in the HLA specification: federates exist as simulation resources with complete transparency to each other. While current RTI implementations limit the distribution of federates, this is an implementation decision and not a requirement of the HLA. The set of objects and interactions available to a federation are defined in its FOM, and object and ownership management allow instances to be shared and divested between running resources without regard to federate location. Synchronization issues can be managed through the judicious use of the time management functions. One shortcoming of the HLA that could prevent a Grid services implementation is the lack of an explicit authentication mechanism. Any federate can join a federation, send any interactions it publishes, receive any attribute updates or interactions it subscribes to, and update any attributes for owned object. In addition, updates and interactions do not carry any explicit identification of their origin, leaving no recourse for application accounting within the specification. This does not necessarily preclude implementing the HLA over Grid services, but it would place severe constraints on other simultaneous uses of the underlying Grid.

6 Conclusions

The XMSF project is only two years old and already has made major progress by defining the role of Web services for interoperation of simulations and demonstrating their effectiveness in multiple projects. The rapid success of these efforts and lack of any major problems in their implementation is highly encouraging. XMSF is able to support interoperation among disparate software systems, including HLA federations, on heterogeneous platforms. The principles of Web services are also applicable in the Grid computing environment where they promise to allow further expansion of the range of distributed simulation possibilities, via Grid services.

Acknowledgement. Work described in this paper was supported in part by the US Defense Modeling and Simulation Office (DMSO) and JFCOM.

References

1. Donald Brutzman, Michael Zyda, J. Mark Pullen and Katherine L. Morse: Extensible Modeling and Simulation Framework (XMSF): Challenges for Web-Based Modeling and Simulation, Findings and Recommendations Report of the XMSF Technical Challenges Workshop and Strategic Opportunities Symposium, October 2002, http://www.movesinstitute.org/xmsf/XmsfWorkshopSymposiumReportOctober2002.pdf
2. W3C (2002) Web Services Architecture Requirements, November 14 2002, Online documents at http://www.w3.org/TR/wsa-reqs.
3. IEEE 1516, Standard for Modeling & Simulation - High Level Architecture.
4. Object Management Group (OMG) website: Model Driven Architecture (MDA); http://www.omg.org/mda/
5. Andreas Tolk, James Muguira: The Levels of Conceptual Interoperability Model (LCIM). Paper 03F-SIW-007; *Proceedings of the IEEE Fall 2003 Simulation Interoperability Workshop*, 2003; http://www.odu.edu/engr/vmasc/TolkPublications.shtml
6. Mikel Petty *et al*: Findings and Recommendations from the 2003 Composable Mission Space Environments Workshop, Paper 04S-SIW-050, *Proceedings of the IEEE 2004 Spring Simulation Interoperability Workshop*
7. Andreas Tolk: Avoiding another Green Elephant, A Proposal for the Next Generation HLA based on the Model Driven Architecture. Paper 02F-SIW-004, *Proceedings of the IEEE 2002 Fall Simulation Interoperability Workshop*, 2002; http://www.odu.edu/engr/vmasc/TolkPublications.shtml
8. Katherine L. Morse, Ryan Brunton and David Drake: Web Enabling an RTI – an XMSF Profile, Paper 03E-SIW-046, *Proceedings of the IEEE 2003 European Simulation Interoperability Workshop*, 2003.
9. Scott Carey, Martin Kleiner, Michael Hieb, and Richard Brown: Standardizing Battle Management Language - A Vital Move Towards the Army Transformation, Paper 01FSIW-067, *Proceedings of the IEEE 2001 Fall Simulation Interoperability Workshop*, 2001.
10. Scott Carey, Martin Kleiner, Michael Hieb, and Richard Brown: Standardizing Battle Management Language - Facilitating Coalition Interoperability, Paper 02E-SIW-005, *Proceedings of the IEEE 2002 European Simulation Interoperability Workshop*, 2002.
11. William Sudnikovich, Michael Hieb, Martin Kleiner: Developing the Army's Battle Management Language Prototype Environment, Paper 04S-SIW-115, *Proceedings of the IEEE 2004 Spring Simulation Interoperability Workshop*, 2004.

12. Michael Hieb, William Sudnikovich, Andreas Tolk, and J. Mark Pullen: Developing Battle Management Language into a Web Service, Paper 04S-SIW-113, *Proceedings of the IEEE 2004 Spring Simulation Interoperability Workshop*, 2004.
13. Multilateral Interoperability Programme (MIP): The C2 Information Exchange Data Model (C2IEDM). 20 November 2003, approved by MIP in Greding, Germany, http://www.mipsite.org/MIP_Specifications/Baseline_2.0/C2IEDMC2_Information_Exchange_Data_Model/t
14. Wikipedia: Grid computing, January 4 2004, Online documents at http://en2.wikipedia.org/wiki/Grid_computing.
15. Rajkumar Buyya: Grid Computing Info Centre, January 29, 2004, Online documents at http://www.gridcomputing.com.