HLA Evolved – A Summary of Major Technical Improvements

Björn Möller
Katherine L Morse
Mike Lightner
Reed Little
Robert Lutz

bjorn.moller@pitch.se
katherine.l.morse@saic.com
MLightner@aegistg.com
little@sei.cmu.edu
robert.lutz@jhuapl.edu

Keywords:
HLA Evolved, FOM Modules, Web Services, Fault Tolerance, DLC, RTI

ABSTRACT: *"HLA Evolved" is the code name for the new version of the High Level Architecture that is expected to be completed in late 2008. This version gives developers and users of distributed simulation a large number of new development, deployment and net centric capabilities. While the new HLA version contains hundreds of updates, this paper focuses on the major technical improvements. Some comments on migration are also given. This paper looks at the following topics:*

- *Modular federation object models (FOMs) and simulation object models (SOMs) make it possible to maintain HLA object models in a more flexible and scalable way, to separate local extensions from standardized models, to support partial federation agreements and even to gradually extend the object model of long-running federations.*
- *Web Services support through the new Web Services Description Language application program interface (WSDL API) gives users of Web Services communication frameworks access to the full HLA functionality over wide area networks while providing features like authentication and encryption. It also provides support for a large number of programming languages.*
- *Fault tolerance support gives federations a well-defined semantics for handling unreliable or crashing federates and network links that go down.*
- *Smart update rate reduction makes it possible to subscribe to the same information with higher or lower update rates. This gives the ability to focus on certain entities or to reduce network traffic on wide area networks.*
- *Dynamic link compatibility makes it easier to switch between different runtime infrastructure (RTI) implementations without modifying, recompiling or re-linking federate code.*

Some discussion is also provided on other improvements such as flexible transportation types, standardized time representations, enumeration of federations, encoding helpers and Object Model Template (OMT) XML aspects.

Most of the above updates are simply extensions to the existing IEEE 1516 functionality. The effort to migrate a federate mainly consists of migrating to the updated API and data types. However, to really take advantage of some new features, for example fault tolerance, some new behavior will need to be added to the federate code.

1. Introduction

This paper seeks to give a summary of major technical improvements in HLA Evolved. It addresses the main types of updates and additions that have been made and the new functionality provided. It also provides a first summary of what you need to do to migrate federates and federations to HLA Evolved. The standards development process, which follows the SISO BPDSP [1] has been covered in an earlier paper [2].

1.1 Technical development history of HLA

The High Level Architecture (HLA) was developed in the early ‘90s with the objective to increase interoperability and reuse for simulations within the U.S. Department of Defense (DoD). Based on a broad objective, stated in the U.S. DoD M&S Vision [3], a variety of solutions were investigated and the basic HLA principles were established. Based on several “proto-federation” experiences, the HLA was refined, resulting in the DoD HLA 1.3 [4] standard that was released in 1998. As the version number “1.3” clearly indicates, it took several steps of refinement to reach a specification that achieved both the required functionality and the necessary technical quality.

Distributed simulation is a constantly evolving field, which means there is a continuing need for a supporting standard to meet the field’s evolving requirement. HLA has been evolving to meet that need. When it was taken to IEEE to become an open international standard, called IEEE 1516 [5], in the year 2000, additional improvements were made, most notably in the Data Distribution Management (DDM) area and the introduction of XML- [6] based formats for object modeling. The HLA standard has achieved widespread international acceptance and use in the last eight years. The experiences of the user community with the 2000 version are reflected in the updates to the specifications developed under the “HLA Evolved” [2] effort.

1.2 HLA and RTI implementations

While HLA is an architecture, its most well-known software component is the runtime infrastructure, RTI. A rough estimate is that there have been 25 major RTI implementation projects around the world. These can be roughly divided as follows: a handful of commercial off-the-shelf (COTS) implementations; ten other, well-maintained implementations; and ten significant but less well-maintained implementations. In addition, there have probably been hundreds of student RTI projects.

These implementations have served many purposes resulting in COTS and government off-the-shelf (GOTS) products as well as academic research results. They have also supported experimentation. They have shown how HLA can be implemented and optimized for different purposes. Insight and HLA competency have been developed within thousands of organizations. Perhaps most importantly is the fact that the HLA standard has been implemented by different government-sponsored organizations across the world, creating widespread buy-in.

In addition to this, a huge number of HLA-compliant products or toolkits have also been developed as COTS, GOTS and in-house projects. Today, HLA-compliant platform simulators, computer-generated forces (CGFs), data loggers, middleware generators, visualization tools, etc., are readily available in a global marketplace.

1.3 Maturity and performance of RTIs

Today’s RTI implementations are considerably more mature than early implementations. While early experimental RTIs had very limited ease-of-use, robustness and performance, a competitive marketplace has changed this. RTIs with easy-to-use graphical user interfaces, development, debugging and tuning tools are now readily available. Robustness and performance have improved considerably. Together with the performance enhancements in CPU technology, this provides more and more power to the simulationist every year. Examples of performance figures on regular Microsoft Corporation’s Windows® computers, based on publicly available figures from COTS RTI vendors are:

Update rate: RTIs can provide 50,000, sometimes more than 100,000 updates of 100 bytes per second between two hosts on a local area network (LAN).

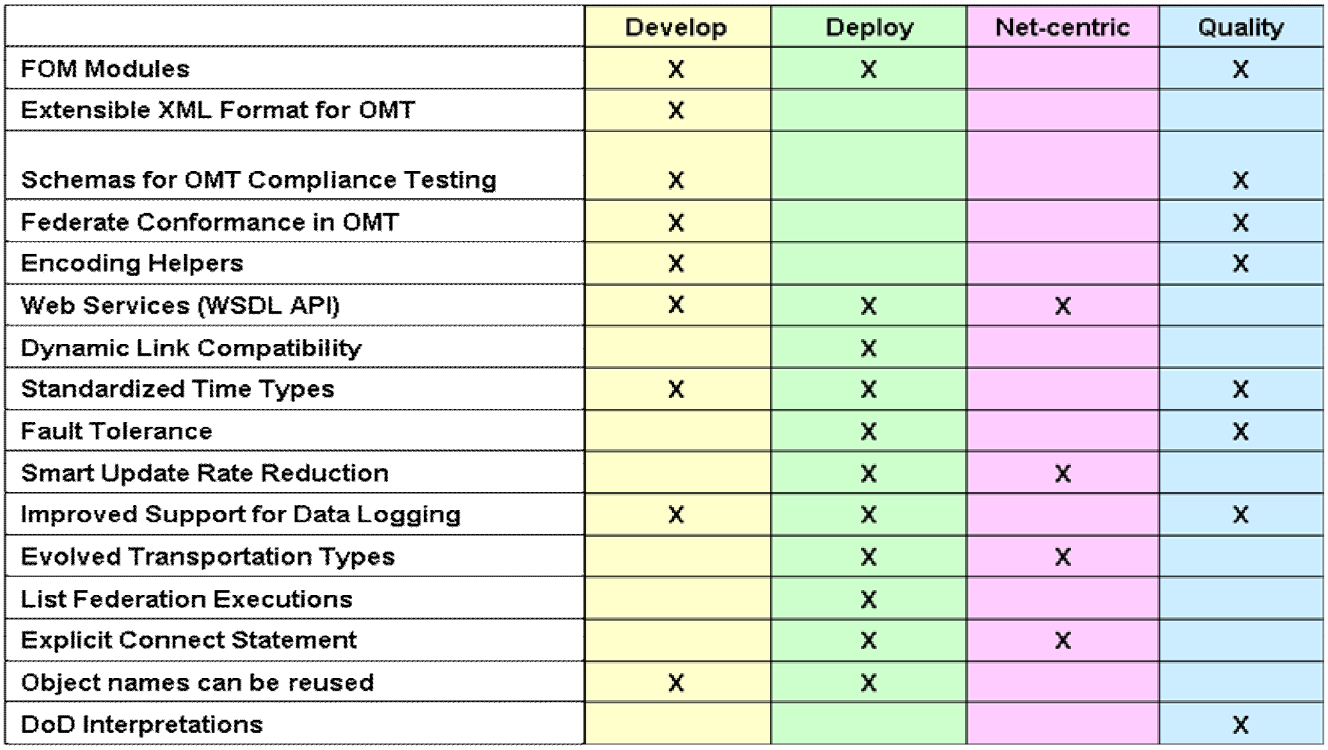
Latency: Updates can be provided between two computers on a LAN with a latency of less than 130 microseconds.

Federation scalability: Federations of hundreds of federates/computers can be supported.

Scenario scalability: Scenarios with 100,000 or more simulated entities are supported, with only the update rate (previously described) limiting the simulation rate.

It is, however, important to understand that federation performance depends on several other components during execution. Typical things that may slow down the speed of a simulation are: the speed at which participating federates can consume or produce updates, how often they process incoming information, the latency of the network, how quickly they perform their calculation before advancing to the next time step, etc.

It should also be noted that speed and scalability are only two of many optimization criteria. Robustness, ease-of-use, security and compatibility with technical environments are examples of other important criteria.



	Develop	Deploy	Net-centric	Quality
FOM Modules	X	X		X
Extensible XML Format for OMT	X			
Schemas for OMT Compliance Testing	X			X
Federate Conformance in OMT	X			X
Encoding Helpers	X			X
Web Services (WSDL API)	X	X	X	
Dynamic Link Compatibility		X		
Standardized Time Types	X	X		X
Fault Tolerance		X		X
Smart Update Rate Reduction		X	X	
Improved Support for Data Logging	X	X		X
Evolved Transportation Types		X	X	
List Federation Executions		X		
Explicit Connect Statement		X	X	
Object names can be reused	X	X		
DoD Interpretations				X

Table 1: New HLA functionality by category

1.4 Types of technical improvements

There are several reasons that we now see a number of technical improvements. Simulation users request new capabilities. HLA must meet new requirements, for example, to be useful in more challenging environments, both from a development and a runtime perspective. The existing standard may also have limitations or problems in the current functionality. It may sometimes be desirable to further align the standard with other standards. There may also be areas where the standard needs to be further clarified.

Table 1 summarizes the HLA Evolved work. The new functionality has been divided into the following categories:

Development features that makes it easier, faster, and less error-prone to develop federates and federations or that provide new features.

Deployment features that improve how and where federates and federations can be deployed.

Net-centric features that facilitate the way federations can be used in net-centric applications.

Quality enhancements that either improve the quality and reduce any ambiguities in the standard or facilitatesthe development of high-quality federates.

The following sections of this paper will walk through major technical changes in the HLA Evolved standards. Suggestions for additional reading are provided. Some details in these papers may have changed after they were produced. The final and authoritative source in all cases is, of course, the finalized HLA Evolved standard. The Simulation Interoperability Standards Organization (SISO) HLA Evolved reflector may also be useful in some cases to gain a deeper understanding of the background of some changes to the standard.

2. Modular FOMs

Problem: Development and reuse of federation object models (FOMs) and simulation object models (SOMs) is difficult, based on the current monolithic approach. Hundreds of projects have adapted standard FOMs to their unique requirements, making it very hard to separate out local extensions. New concepts cannot be introduced in an already running federation.

Solution: The FOM can now be provided as modules that contain a subset of the FOM, for example, selected object classes or data types. Different aspects of a problem, for example, platforms, sensors and communications and reusable data types, can be developed and described in different modules. Locally developed modules can extend standardized modules. New modules (and thereby concepts) can be introduced into an already running federation.

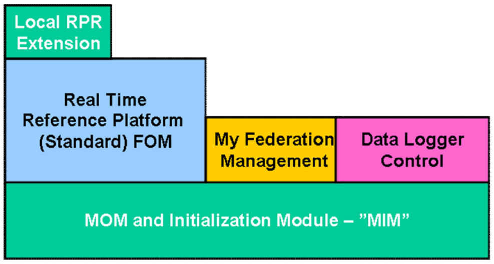


Figure 1: Sample FOM module hierarchy

All of the pre-defined concepts in the HLA FOM, such as management object model (MOM) and predefined data types, have been moved to a separate module called the MOM and Initialization Module (MIM). This makes the modules smaller and easier to handle for developers.

Figure 1 shows a sample FOM module hierarchy where the FOM is composed of the standard MIM, a standardized reference FOM, an extension to the reference FOM, a module for federation management and a module for data logger control.

A new set of services makes it possible for a federate to inspect which FOM modules have been loaded and even to inspect the content of them.

Additional reading: Further details and practical use cases are provided in [7] and [8]. Modular FOMs can be effectively developed using base object models (BOMs) [9], which are described in [10].

3. Web Services API (WSDL)

Problem: It is difficult to provide an HLA-based simulation as a service, for example, over the Internet. Web Services is the preferred way to interoperate in some "neighboring" communities.

Solution: HLA now provides an application program interface (API) based on the Web Services Description Language (WSDL) [11]. While technically WSDL is really a protocol description, it still provides exactly the same functionality as the C++ and Sun Microsystems' Java™ APIs. A federate can now connect using Web Services across LANs and WANs, optionally using https-based encryption and authentication.

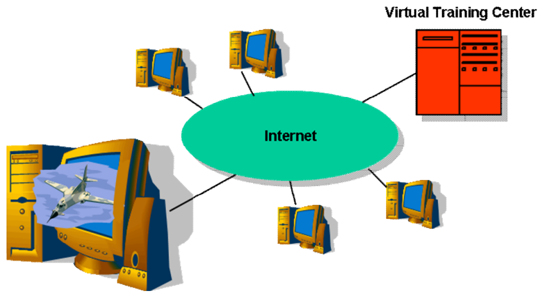


Figure 2: Simulation as a service

Figure 2 illustrates how a server on a known address on the Internet can support HLA-based interoperability with clients in many different locations. The principle is basically the same as for a web server and a web browser.

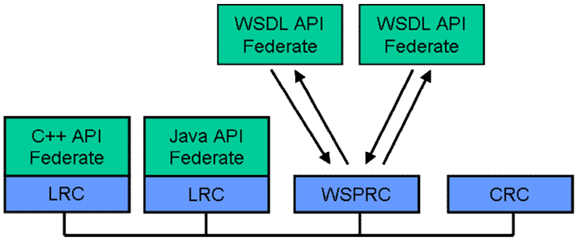


Figure 3: RTI with Web Service Provider

To support the Web Services API, an RTI needs to provide a Web Services Provider RTI Component (WSPRC) that one or more WSDL federates can connect to using a URL. This is shown in Figure 3.

WSDL is supported for a wide range of programming languages, including C, C++, Java, Fortran, ADA, Perl, and many more.

Additional reading: An overview can be found in [12], suggestions for practical applications in [13] and some more detail in [14]. The first prototype for a WSDL API is described in [15] and some related work in [16].

4. Fault Tolerance

Problem: As HLA federations have moved from well-controlled labs into real-life environments, they need to operate in less-than-perfect environments with unreliable networks. As federation size grows, so does the likelihood that some computers or components will fail. As the number of organizations that contribute to each federation grows, it is more and more difficult to control the quality of each federate.

Solution: HLA now provides a well-defined mechanism whereby failing federates can be removed from the federation. There is also a clear mechanism for signaling the faults both to the federation that lost one or more federates and to a federate that lost the connection to the federation. Figure 4 illustrates these signals.

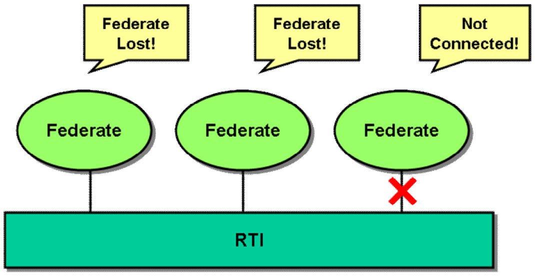


Figure 4: Fault signaling

It is possible for federates to gain insight into the current state of the federation. This makes it possible to evaluate, manually or automatically, the degree of degradation.

Additional reading: A more detailed description, together with suggested design patterns for fault tolerant federations, is provided in [17] with additional background information available in [18] and [19].

5. Smart Update Rate Reduction

Problem: Different federates may need to get updates for the same information but with different update rates. Common use cases are when data produced by a high-fidelity simulator, like a flight simulator, needs to be consumed by a simulator that monitors a wide range of objects at a lower data rate, like a command and control system. It is also important in order to be able to reuse older, slower federates in a new context with higher update rates or larger scenarios. Another problem is when you have limited data link capacity for some sites and need to reduce the data flow.

Solution: HLA now offers an option to subscribe to attribute updates with a specified maximum update rate. Figure 5 illustrates how the update rate requirements are communicated.

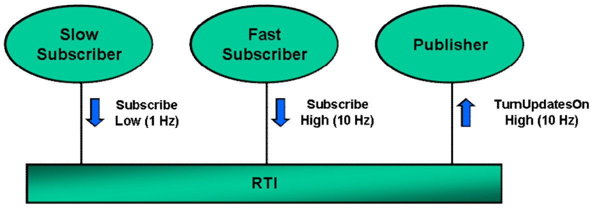


Figure 5: Subscribing with different update rates

The corresponding publisher of data gets a notification of the maximum rate at which an attribute of an instance is subscribed to. It is even possible to combine this with Data Distribution Management (DDM) to subscribe to the data of some instances (like "active threat") with a higher update rate and some other (like "remote friendly platforms") with a lower update rate.

Additional reading: A more detailed description is found in [20].

6. Dynamic Link Compatibility

Problem: There have been some challenges with earlier versions of the standard with executing the same federate with different RTIs.

Solution: Based on an earlier SISO product, called the Dynamic Link Compatibility API ("DLC API") [22] [23], an HLA Evolved version of the DLC API has been developed. We suggest the informal term "Evolved DLC API" (EDLC API) to distinguish the Evolved version from earlier DLC APIs.

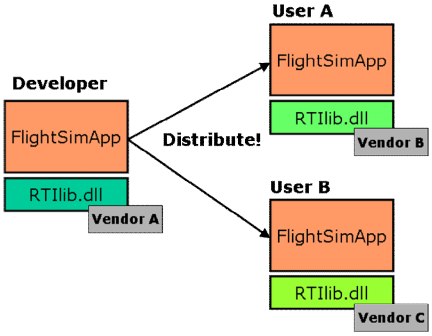


Figure 6: Dynamic Link Compatibility

This makes it possible to move the same federate between different RTI implementations. Figure 6 illustrates how a federate can be developed using one RTI implementation and then be distributed for use with several other RTI implementations. In case a user has installed several RTIs on their computer, the required RTI can be dynamically selected.

Note that there are two related features that aren't directly part of the EDLC API but that may be very important for a user who wants to switch between different RTI implementations:

Standardized time types, whereby the RTI is required to always provide one standardized integer-based and one float-based time implementation. This makes a federate less dependent on time implementations provided by one particular RTI supplier. As a side effect, it also makes it easier to use HLA Time Management.

Evolved transportation types, where the existing mechanism for RTI-specific transportation types has been enhanced. In case a requested transportation type isn't provided by the currently used RTI implementation, there is a clear scheme for falling back to one of the standardized types.

7. More Interface Specification Improvements

There is a plethora of additional enhancements in HLA Evolved. Some major features worth mentioning are:

Improved data logging support, where it is now possible for a data logger to focus on data produced by one particular “producing federate.”

Encoding helpers, which provide marshalling (serialization) and de-marshaling of data that is to be sent over the RTI. This is expected to reduce time and risk in integration. Additional reading is provided in [21].

Federate names, whereby each federate now has a unique name in a federation. The name is provided when the federate joins.

Reusable object names, which makes it possible to reuse names of objects that have been deleted. This makes it easier for a federate to re-join a federation or for a data logger to play back data.

Connect and List services, whereby a federate can first connect to an RTI (with certain connection parameters) and then list available federation executions.

Save/Restore operations now provide more information, namely a federates’ pre/post restore ID. It is also possible to abort a save and restore operation.

Synchronization Points can not only be achieved, but a federate can also achieve them with a “false” flag, ultimately resulting in a “federation-synchronized” state with a list of the federates that raised this flag.

Ownership now provides a “release denied” service that a federate can call in response to a request attribute ownership release.

Concurrency/Reentrancy, where it has been clarified exactly when the RTI Ambassador and Federate Ambassador can be called. It is now possible for a federate to call most RTI services during a callback.

Callback models: A federate can now choose between two clearly defined callback models, namely *Evoked* (“ticked”) and *Immediate* delivery.

Early or late evaluation, whereby a federate may select whether subscriptions are evaluated at send time or delivery time.

Object deletion can now be time-managed.

In addition the 210 **DoD interpretations** that were collected for the HLA 1516-2000 have been incorporated into the standard.

8. Technical Updates in OMT

One major update that affects both the HLA OMT specification and the interface specification is the support for **Modular FOMs** described earlier. The biggest impact on the standard is the rules for merging FOM Modules. In most cases, it means merging the content of the modules; for example, taking a union of all object classes and data types provided. In some particular cases, like switches and time representation, the content, if provided, must be equal.

Another new feature is the **extensible XML format** which allows users to add extra data to a FOM or SOM in the form of new XML elements.

There are also **three new XML schemas** for compliance testing of FOMs and SOMs:

The *DIF format schema* tests that the basic format (syntax) is followed.

The *FDD schema* tests that the FDD part of a FOM can be used for initializing a federation; that is, it can be loaded into an RTI. Note that this schema is actually part of the interface specification since it relates to the RTI.

The *OMT Compliance Schema* tests that the content is correct and consistent (with some limitations).

Note that a FOM module can only be required to adhere to the DIF schema (since it may exclude some tables and depend on other modules) whereas an assembled FOM must adhere to at least the FDD schema to be useful for an RTI, and ultimately, the OMT schema.

The **identification table** has been extended with new metadata fields.

There is also an option to provide a **glyph (icon)** which can be used by OMT tools to provide a graphical representation of a FOM, SOM or module.

For federate developers that want to send their federates for federate compliance testing [24], it is now possible to describe their RTI **services usage** directly in the SOM. This is part of a new OMT "Conformance" section, which also identifies formal tests for each OMT table to verify conformance.

There are also **three new switches** in the FOM to control the runtime behavior of the RTI.

9. Migration Strategies

This section focuses on migration to native HLA Evolved, whereas bridging may, in some cases, be a viable intermediate solution.

As of today a relatively small number of federates have been migrated to early releases of HLA Evolved RTIs. Some early experiences are provided here and more is expected during the upcoming years.

The first step would be to convert the FOM from HLA 1.3 or HLA 1516-2000 to the HLA Evolved format. This can largely be done using automated tools. At this time, you may optionally want to split the FOM into modules. You may also want to review the values of the new switches.

Federations that currently use HLA 1.3 will need to look at the HLA 1.3 to 1516 migration step that is covered elsewhere [25]. Major issues to consider here are that subscriptions now are additive and the re-design of DDM, if used.

For the federate code, developers will also need to adapt to the new API. No semantics changed from 1516-2000, but some new features have been added, and most data types have changed. Handles that were integers in HLA 1.3 (link-compatible but not type-safe) became handles that were type-safe but not link-compatible in HLA 1516-2000. The EDLC APIs introduce data types that are both link-compatible and type-safe.

Also note that a number of optional parameters in some calls have been moved into separate data objects, for example SupplementalReflectInfo.

If you don't have a powerful data-marshalling library of your own, you may consider using the encoding helpers. If your federate uses time management, you may also consider using the new standardized time representations.

The most significant semantic change for most federates is if they want to implement some level of fault tolerance. Read [17] on different design patterns. We suggest using the pattern "The required federation subset" for analysis applications and "The optional federation" for training applications as a starting point for further design discussions. For some federations, this means that they may need to consider "late joiner" issues where the HLA request/provide pattern for attribute values is a common solution.

The most interesting part of the migration may be if you want to use any of the new features like Smart Update Rate Reduction.

Federations that start afresh will, of course, have a pleasant situation where they can simply dig into the smorgasbord of solutions provided by the HLA Evolved standard, with Modular FOMs as the natural starting point.

10. Summary

As HLA has been applied in a large number of federations all over the world, experiences have been collected and fed back into the HLA standards.

As technology evolves and requirements expand, solutions based on this have also been fed back into the HLA standards development process. The result is the HLA Evolved standards that provide even more features to better meet the needs of modeling and simulation communities such as training, analysis, testing, acquisition and experimentation.

HLA vendors have already committed to implementing the new HLA Evolved standards, and early experimental HLA Evolved federations have already been demonstrated.

Finally it should be pointed out that HLA Evolved is a good example of how an open standards process can be responsive to the needs of developers and users. In particular it has provided a forum for users with experiences of existing architectures where problems have been solved, differences have been mediated and new capabilities have been added.

References

[1] SISO, "SISO Balloted Products Development and Support Process (BPDSP)", SISO-ADM-003-2008, SISO, 17 Jan 2008.

[2] Roy Scrudder, Gary M. Lightner, Robert Lutz, Randy Saunders, Reed Little, Katherine L. Morse, Björn Möller. "Evolving the High Level Architecture for Modeling and Simulation". Proceedings of the 2005 Interservice/Industry Training, Simulation & Education Conference, Paper No. 2157, National Training Systems Association, December 2005.

[3] US DoD M&S Vision, DoD Executive Council on Modeling and Simulation (EXCIMS), March 13, 1992

[4] "High Level Architecture Version 1.3", DMSO, www.dmso.mil, April 1998

[5] IEEE: "IEEE 1516, High Level Architecture (HLA)", www.ieee.org, March 2001.

[6] Bray, Tim; Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau (September 2006). "Extensible Markup Language (XML) 1.0 (Fourth Edition) - Origin and Goals" World Wide Web Consortium, http://www.w3.org/TR/2006/REC-xml-20060816/

[7] Björn Möller, Björn Löfstrand, Mikael Karlsson. "An Overview of the HLA Evolved Modular FOMs", Proceedings of 2007 Spring Simulation Interoperability Workshop, 07S-SIW-108, Simulation Interoperability Standards Organization, March 2007.

[8] Björn Möller, Björn Löfstrand. "Use Cases for the HLA Evolved Modular FOMs", Proceedings of 2007 Euro Simulation Interoperability Workshop, 07E-SIW-040, Simulation Interoperability Standards Organization, June 2007. Download here

[9] SISO, "BOM Template Specification", SISO-STD-003-2006, SISO, 31 March 2006.

[10] Björn Möller, Paul Gustavson, Bob Lutz, Björn Löfstrand. “Making Your BOMs and FOM Modules Play Together”, Proceedings of 2007 Fall Simulation Interoperability Workshop, 07F-SIW-069, Simulation Interoperability Standards Organization, September 2007

[11] W3C: “Web Services Description Language (WSDL) 1.1”, W3C, URL: www.w3.org/TR/wsdl

[12] Björn Möller, Staffan Löf. “A Management Overview of the HLA Evolved Web Service API”, Proceedings of 2006 Fall Simulation Interoperability Workshop, 06F-SIW-024, Simulation Interoperability Standards Organization, September 2006.

[13] Björn Möller, Clarence Dahlin, Mikael Karlsson. “Developing Web Centric Federates and Federations using the HLA Evolved Web Services API”, Proceedings of 2007 Spring Simulation Interoperability Workshop, 07S-SIW-107, Simulation Interoperability Standards Organization, March 2007.

[14] Björn Möller, Clarence Dahlin. A First Look at the HLA Evolved Web Service API. Proceedings of 2005 Euro Simulation Interoperability Workshop, 06E-SIW-061, Simulation Interoperability Standards Organization, June 2006.

[15] Katherine L. Morse, Ryan Brunton and David Drake, "Web Enabling an RTI - an XMSF Profile", Proceedings of the 2003 European Simulation Interoperability Workshop, June 2003

[16] Björn Möller, Staffan Löf. “Mixing Service Oriented and High Level Architectures in Support of the GIG” Proceedings of 2005 Spring Simulation Interoperability Workshop, 05S-SIW-064, Simulation Interoperability Standards Organization, April 2005.

[17] Björn Möller, Björn Löfstrand, Mikael Karlsson. ”Developing Fault Tolerant Federations using HLA Evolved” Proceedings of 2005 Spring Simulation Interoperability Workshop, 05S-SIW-048, Simulation Interoperability Standards Organization, April 2005.

[18] Trevor Pearce, Björn Möller. “Towards Fault Tolerant RTIs, Federates and Federations.” Proceedings of 2005 Spring Simulation Interoperability Workshop, 05S-SIW-129, Simulation Interoperability Standards Organization, April 2005.

[19] Björn Möller, Mikael Karlsson. “Developing the Fault Tolerance Support Extensions for HLA” Evolved. Proceedings of 2005 Euro Simulation Interoperability Workshop, 05E-SIW-019, Simulation Interoperability Standards Organization, June 2005.

[20] Björn Möller, Mikael Karlsson. “Developing well-balanced federations using the HLA Evolved smart update rate reduction”. Proceedings of 2005 Fall Simulation Interoperability Workshop, 05F-SIW-87, Simulation Interoperability Standards Organization, September 2005.

[21] Björn Möller, Mikael Karlsson, Björn Löfstrand. ”Reducing Integration Time and Risk with the HLA Evolved Encoding Helpers”. Proceedings of 2006 Spring Simulation Interoperability Workshop, 06S-SIW-042, Simulation Interoperability Standards Organization, April 2006.

[22] SISO: “Dynamic Link Compatible HLA API Standard for the HLA Interface Specification” (IEEE 1516.1 Version), (SISO-STD-004.1-2004)

[23] Len Granowetter, “Design of the Dynamic-Link-Compatible C++ RTI API for IEEE 1516”, Proceedings of 2004 Fall Simulation Interoperability Workshop, .04F-SIW-086, Simulation Interoperability Standards Organization, September 2004

[24] J-L. Igarza, R. Mauget, M. Adelantado, R. Saunders, R. Scrudder, M. Crooks, “NATO and NATO/PfP Nations HLA Compliance Certification: History, Lessons Learned and Proposed Enhancements”, Proceedings of the 2007 Fall Simulation Interoperability Workshop, 07F-SIW-075, Simulation Interoperability Standards Organization, September 2007.

[25] HLA 1.3 to 1516 Migration Guides, http://www.msco.mil (new posting coming soon)

Author Biographies

BJÖRN MÖLLER is the vice president and co-founder of Pitch, the leading supplier of tools for HLA 1516 and HLA 1.3. He leads the strategic development of Pitch HLA products. He serves on several HLA standards and working groups and has a wide international contact network in simulation interoperability. He has twenty years of experience in high-tech R&D companies, with an international profile in areas such as modeling and simulation, artificial intelligence and Web-based collaboration. Björn Möller holds an MSc in computer science and technology after studies at Linköping University, Sweden, and Imperial College, London. He is currently serving as the vice chairman of the SISO HLA Evolved Product Development Group.

KATHERINE L. MORSE, Ph.D., is a technical fellow at SAIC. She received her B.S. in mathematics (1982), B.A. in Russian (1983), M.S. in computer science (1986) from the University of Arizona, and M.S. (1995) and Ph.D. (2000) in information and computer science from the University of California, Irvine. Dr. Morse has worked in the computing industry for almost 30 years, specializing in the areas of simulation, computer security, compilers, operating systems, neural networks, speech recognition, image processing, and engineering process development. Dr. Morse is the primary author of the HLA DDM services, and has served as an officer and drafting group member of the original and current IEEE working group for the HLA. She is the 2007 winner of the IEEE Hans Karlsson Award for leadership in development of M&S standards and exemplary collaboration.

GARY "MIKE" LIGHTNER is a senior simulation engineer at The AEgis Technologies Groups Inc., where he serves as director of the Simulation Technologies and Advanced Research Group. He holds a B.S. degree in computer science from the College of Engineering, Arizona State University, and an M.S. degree in computer science from the Air Force Institute of Technology. With over 20 years of experience in computer and modeling and simulation (M&S) systems design, development, implementation and advanced technology R&D, he establishes, communicates and fosters the vision, goals and agenda for researching, developing and advancing new and advanced M&S technologies, techniques and standards for the AEgis Technologies Group. In addition, he currently provides R&D and technical expertise to the Defense Modeling and Simulation Office (DMSO) HLA, VV&A and Education programs as well as the Canadian Synthetic Environment Coordination Office (SECO). Mr. Lightner was the drafting editor of the IEEE Std 1516-2000 IEEE Standard for Modeling and Simulation High Level Architecture (HLA) – Framework and Rules and the chair of the Ballot Resolution and IEEE and DoD Interpretations committees. He is currently serving as the drafting editor for the IEEE Std 1516-200X specification under the SISO HLA Evolved Product Development Group.

REED LITTLE is a senior member of the technical staff at the Software Engineering Institute, Carnegie Mellon University. He investigates software architecture representation issues, especially the relationship to and support of software product lines. Little was the drafting editor of the IEEE Std 1516.1-2000 IEEE Standard for Modeling and Simulation High Level Architecture (HLA) – Federate Interface Specification and as well as the chair of the Ballot Resolution committee. He is currently serving as the drafting editor for the IEEE Std 1516.1-200X Federate Interface Specification as part of the SISO HLA Evolved Product Development Group.

ROBERT LUTZ is a principal staff scientist at The Johns Hopkins University Applied Physics Laboratory (JHU/APL). He has over 27 years of experience in the design, implementation, and evaluation of computer modeling and simulation (M&S) systems for military customers. He received his M.S. degree in operations research from the State University of New York at Stony Brook in 1980. Since joining JHU/APL in 1992, Mr. Lutz has assumed leadership roles on several M&S programs, including the Naval Simulation System (NSS), Joint Warfare System (JWARS), and the Simulation Based Acquisition (SBA) initiative. Mr. Lutz also served as the technical editor for IEEE 1516.2 (HLA Object Model Template) and as working group chair for IEEE 1516.3 (HLA FEDEP). Currently, he is supporting a variety of M&S initiatives in support of the U.S. Modeling and Simulation Coordination Office (MSCO), the U.S. Joint Forces Command (USJFCOM), and the Navy's Broad Area Maritime Surveillance (BAMS) Program. He also serves as a guest lecturer in The Johns Hopkins University Whiting School of Engineering.