Lessons Learned in Applying the HLA to the Joint Training Confederation

Zach Furness
Mark A. Robinson
The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102-3481
703-983-6614, 703-983-5401
zfurness@mitre.org, wrobinso@mitre.org

Carol Chiang
MIT Lincoln Laboratory
244 Wood Street
Lexington, MA 02420-9185
781-981-4009
chiang@ll.mit.edu

John Logsdon
Mitchell Primas
U.S. Army Simulation Training and Instrumentation Command (STRICOM)
ATTN: AMSTI-PM WARSIM
12350 Research Parkway,
Orlando, FL 32826-3276
407-384-3622, 407-384-3850
john_logsdon@stricom.army.mil, mitchell_primas@stricom.army.mil

Keywords:
JTC, HLA, RTI, ALSP, JIS, ADAPTOR, FMT

ABSTRACT: *The Joint Training Confederation (JTC) is a collection of service models that have been used to support joint training exercises for Joint Commanders and their staffs. Since 1992, interoperability between these models has been achieved through the application of the Aggregate Level Simulation Protocol (ALSP). With the advent of the High Level Architecture (HLA), it became desirable to replace ALSP with a combination of the Run Time Infrastructure (RTI) and an "ADAPTOR" that would provide an interface to JTC models without having to modify the existing ALSP interfaces of each individual model. In addition, a new Federation Management Tool (FMT) was developed that significantly improves the ability to monitor the JTC during exercises and correct problems as they develop. This paper will review the lessons learned in design, development, and testing of the RTI, ADAPTOR, and FMT, in the JTC. It will address the functional, performance, and reliability of the RTI, ADAPTOR, and FMT that has been observed during testing applications over the past year, and will look into the longer term applications of these new components in the JTC.*

1. Background

1.1 Overview of the JTC

The Joint Training Confederation (JTC) is composed of a collection of service models that have been used to support joint training exercises since 1992. The models exchange information via the Aggregate Level Simulation Protocol (ALSP), which provides simulation object, time, data, and exercise management services [1]. Since 1993, The MITRE Corporation has acted as the System Engineer for the JTC, with responsibility for development of the ALSP Infrastructure Software (AIS), integration of new simulation capability into the JTC, overall testing of the JTC, and fielding of the JTC each year. The 1999 version of the JTC consists of nine simulations and includes the major functionality for joint training exercises (see Table 1.1). These simulations support combat interactions between air, land, sea and space objects (including tactical ballistic and cruise missiles), provide a common electronic warfare environment, and provide a high resolution

training environment for Army logisticians and intelligence cells.

Table 1.1 1999 JTC Simulations

JTC Simulation	Primary Functionality
CBS (Corps Battle Simulation)	Represents Army combat operations
AWSIM (Air Warfare Simulation)	Represents Air Force combat operations
RESA (Research, Engineering, and Systems Analysis)	Represents Naval combat operations
MTWS (Marine Air Ground Task Force (MAGTF) Tactical Warfare Simulation)	Represents Marine Corps combat operations
CSSTSS (Combat Service Support Training Simulation System)	Represents Army logistics
JQUAD (consists of four related models: JECEWSI - Joint Electronic Combat Electronic Warfare Simulation, JCAS - Joint Command and Control Attack System, JOISIM – Joint Operational Intelligence Simulation, and JNETS – Joint Network Simulation)	Represents Electronic Warfare, Strategic Targets, Joint Intelligence, and Infrastructure Assets
TACSIM (Tactical Simulation)	Represents Intelligence Assets and Information Feeds
MDST (Missile Defense Space Tool)	Represents Satellites, Theater Missile Defense
AMP (Analysis of Mobility Platform)	Represents Military Transportation

1.2 JTC HLA Transition

For the past two years, the JTC Systems Engineer has been developing components that would facilitate the transition from the existing AIS to the High Level Architecture (HLA) Run-time Infrastructure (RTI) whose use has been mandated for interoperability between Department of Defense (DoD) simulations. In addition to the RTI, two other components that are fundamental to the transition are the ALSP Data and Protocol Transfer Over RTI (ADAPTOR) and the Federation Management Tool (FMT). The ADAPTOR translates messages from JTC simulations into RTI service calls (and vice versa) and provides some ALSP functionality not present in the RTI [2]. The FMT is a graphical user interface that displays time and object management information and information on communications infrastructure [3]. Collectively, the ADAPTOR, RTI, and FMT are referred to as the JTC Infrastructure Software (JIS). Figure 1.1 and Figure 1.2 shows the comparisons between the AIS and JIS architectures.

The transition to the use of the RTI is expected to have multiple benefits. First, it will remove the burden of maintaining the AIS from the Systems Engineer by migrating to a broadly supported RTI. Second, it will provide opportunities to leverage standard HLA test and analysis tools for use in the JTC, which has had to rely upon AIS-based tools (provided by the Systems Engineer) for diagnostic purposes. Third, it will allow JTC models that are pursuing HLA migration to attach "directly" to the RTI, without the need to maintain a separate ALSP version of their models. Those models that do not plan to migrate to HLA will still be able to use their ALSP models in the RTI-based JTC, via the ADAPTOR. Finally, the integration of the RTI at JTC user sites prior to fielding of the Joint Simulation System (JSIMS) will provide early use opportunities for building user confidence and expertise in running the RTI. JSIMS currently plans to utilize the RTI for interfaces to C4I systems and external federates. This has several advantages such as: 1) reducing training costs associated with RTI/JSIMS use and 2) reducing JSIMS integration complexity by integrating the RTI well ahead of JSIMS components.

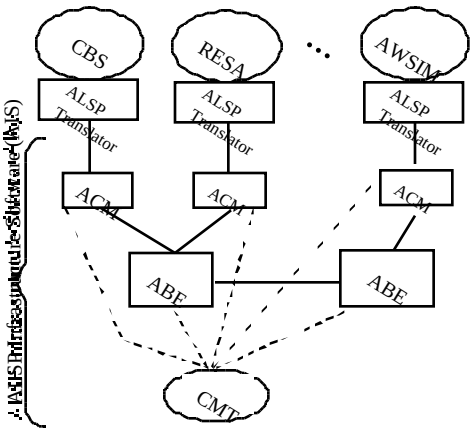

Figure 1.1 AIS Supported JTC

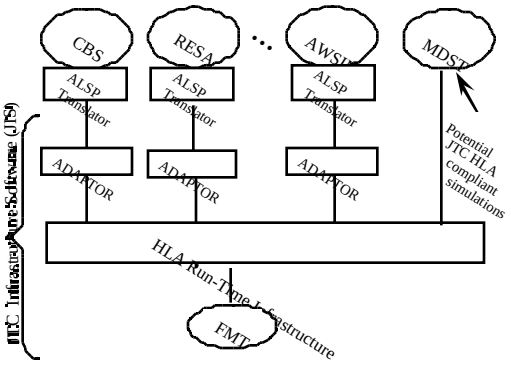

Figure 1.2 JIS Supported JTC

1.3 Transition Plan

Integration and testing of the JIS with the current RTI 1.3 occurred throughout the fall of 1998 and the winter of 1999. Testing consisted of a combination of events within the MITRE lab and external test events at the Joint Training, Analysis, and Simulation Center (JTASC) in Suffolk, VA, the Logistics Exercise & Simulation Directorate (LESD) in Ft. Lee, VA, and the Warrior Preparation Center (WPC) in Einsiedlerhof, Germany. Table 1.2 summarizes the major JIS test events and their purpose:

Table 1.2 Major FY99 JIS Test Events

Date	Place	Test Event	Purpose
Aug 98	JTASC	JIS Functional Test	Test basic functionality with subset of JTC
Sep 98	JTASC	Individual Model Testing	Separate testing of each model interface (CBS, AWSIM, JQUAD, MTWS, RESA, TACSIM, MDST)
Oct 98	JTASC	JIS Performance Test	Test performance of JTC subset
Nov 98	LESD	CBS-CSSTSS Testing	Testing of initial registration of CBS and CSSTSS objects
Dec 98	MITRE	AMP Testing	AMP Testing
Jan 99	WPC	All Actor Integration (AAI)	Full JTC verification test
Feb 99	MITRE	Regression Testing	Regression testing of RTI v1.3, release 6
Mar 99	JTASC	Confederation Test (CT)	Full JTC acceptance test

At the Confederation Test (CT) in March of 1999 (the final test prior to fielding the JTC for 1999) problems with the JIS remained unsolved. Most importantly, the performance of the ADAPTOR and RTI together could not handle the registration of more than 10,000 objects during the CT Load Test. Also, the lack of standard operating procedures (SOPs) for the JIS contributed to additional downtime when attempts to recover the infrastructure were being made. Thus, a decision to revert to the existing AIS software was made during the CT along with subsequent fielding of the AIS for the 1999 JTC.

In June 1999, the ALSP Review Panel agreed to pursue fielding of the JIS in the 2000 version of the JTC if it successfully passes formal JTC testing at next year's CT. With this in mind, current efforts are focused along two paths - continued testing and optimization with RTI 1.3, and testing and integration with RTI 1.3NG. A testing strategy has been developed which will combine testing within the MITRE lab and external sites to ensure that JTC functional, performance, and operational capabilities are met for the year 2000 JTC.

1.4 HLA Functionality of Special Interest to the JTC

Several aspects of the JTC utilize services within the HLA that are not typically used in other applications. Among these are the following:

1.4.1 Unique Object Names for the Life of the Federation

The HLA requires that all registered objects have a unique name for the life of the federation. The AIS requires that each registered object have a unique name for the life of the owning actor[i]. When an actor resigns from an AIS confederation[ii] and deletes its owned objects, the confederation no longer has knowledge of these objects. AIS actors may re-join the confederation and register objects with the identical name as before (prior to resigning), and each object will be discovered by the confederation as a new object.

The RTI provides a unique name for each object if federates do not use the unique name field in the *Register Object Instance* service call. However, the ADAPTOR uses the unique name field for its own internal object identification. To accommodate the unique name requirement in the HLA, the ADAPTOR appends a real-time

[i] An "actor" is the term used to describe a model, or federate, that maintains an ALSP interface

[ii] A confederation is a group of models that all utilize an ALSP interface, analogous to an HLA federation

time stamp to the unique name field whenever a federate registers objects.

1.4.2 Save and Restore Functionality

Two of the more important operations performed by the JTC are save and restore. During a typical JTC exercise scenario the infrastructure may operate for a seven to fourteen day period. Some exercises even require twenty-four hour operation during that period. Since the infrastructure and simulations are distributed, there is a potential for multiple points of failure. The long duration of an exercise, coupled with distributed components, requires a mechanism for re-establishing the state of the infrastructure and simulations (using save and restore). Furthermore, to maintain training realism, the infrastructure and simulations must accomplish this in a timely manner.

1.4.3 Object Discovery

The HLA provides object discovery through the RTI *Discover Object Instance* service callback. JTC federates using the RTI with the ADAPTOR cannot use this callback to discover objects. This is because in the AIS, object discovery is based upon object attribute values. Since there are no attributes associated with the *Discover Object Instance* callback, the ADAPTOR uses the RTI *Reflect Attribute Value* service callback to assist the federate in discovering objects. When the initial *Reflect Attribute Value* callback is called for an object, the ADAPTOR forwards an ALSP *create* message to the federate. This *create* message contains the object identification, attributes, and attribute values received in the callback.

1.4.4 Time Management

The JTC operates in both a time step and event driven mode. Even though the JTC can operate in a variable time step mode, the normal operation is a one minute time step. Each federate asks for time advances (*Time Advance Request*) in increments of one minute and all federates are both regulating (*Time Regulation Enabled*) and constrained (*Time Constrained Enabled*). The entire JTC advances in a locked time step manner with messages being sent and received with a time stamp. In addition to the time advance request, federates can use a relative request service. This service allows federates to request for time advances using time steps relative to their current simulation time. In the case of a one-minute time step a federate would set the relative request parameter to one-minute.

The JTC also uses somewhat unique metrics for measuring the evolution of time during exercises. Of utmost importance is the ability to maintain confederation rate, or the ratio of simulation time to real (wall clock) time. For most training exercises a rate of 1:1 is maintained, if possible. However, pauses during confederation saves, slow processing by actors, and game crashes often require JTC to run faster than 1:1 to "catch-up" to real time. During some test events (the AAI and CT), the JTC will run as fast as 3:1 in order to expedite lengthy and complex test steps.

1.4.5 Attribute Ownership Transfer

In the JTC it is common for federates to transfer attribute ownership properties for specific object instances. The very nature of the JTC being a multi-service federation makes the JIS a federation in which each federate offers a specific functionality (see Table 1). Objects are registered throughout the federation by individual federates and each of these objects can contain attributes which will be updated by different federates. It is the responsibility of each individual federate to update attributes that are included within their specific service [4]. The ADAPTOR utilizes all of the HLA Ownership Management Services to provide attribute ownership functionality across the JTC.

2. JIS Lessons Learned

Test events, such as the CT or AAI, employ all the JTC federates and generate more than 10,000 objects, all updated via reliable transport. Many of these objects also have divested attributes. These characteristics stress aspects of the infrastructure in ways not previously observed in laboratory testing. Unexpected system behavior and inadequate initial performance necessitated an in-depth analysis of RTI API assumptions and infrastructure configuration.

2.1 Bundling

The JTC initially experienced a message-ordering problem due to a bug within RTI 1.3. It was possible for ownership transfer messages for an object to arrive before the Local RTI Component (LRC) had discovered the object. If the ownership transfer message arrived first, the operation would fail. A workaround for this problem was to turn off all forms of message

bundling as this avoided exercising the bug. Unfortunately, unbeknownst to the JTC, turning off bundling also significantly degraded performance. Since the JTC had not been testing in-house with a large federation and could not enable bundling because of the bug, the performance hit was not readily apparent until the AAI in January 1999, which was the first time all JTC models were employed concurrently.

At the AAI, the federation ran very slowly when under heavy load and experienced near 100% CPU usage. It was suspected that usage of bundling could improve performance, but until the message-ordering problem was resolved, enabling bundling was not an option. Near the end of the AAI, a fix for the message-ordering problem was implemented and tested allowing the re-enabling of bundling as well as internal LRC bundling of Distribution Object Manager (DOM) messages. The result was a definite improvement in performance; computers hosting ADAPTOR and RTI components were no longer operating at near 100% CPU usage, and the federation rate also increased. In general, unless there are stringent latency requirements as well as a small number of federation objects, bundling should be enabled whenever possible.

2.2 Reliable Distributor Configuration

In laboratory testing of RTI 1.3, the JTC has perceived considerable performance improvements using the reliable distributor (reldistr) configuration known as the simple network topology [5]. A single reldistr is used by all federates. During the AAI and CT, the JTC used the trivial network topology, which is often referred to as auto-config mode because each federate is automatically configured to have its own internal reldistr. It was discovered, however, that the use of a single reldistr exacerbated an end-to-end flow control problem that occurred when a large amount of reliable traffic was routed. This problem was sporadic and difficult to track down using the internal reldistrs, but much more easily diagnosed using the single reldistr. A fix is being implemented in RTI 1.3 to try to keep this under control so that the JTC can reliably use the single reldistr configuration if it chooses.

While there is not a definitive explanation as to why the single reldistr configuration performs better under certain circumstances, there are several possible reasons as to why that configuration may outperform the auto-config mode. With a single reldistr, each federate sends and receives all of its reliable traffic via this one connection. In an internal reldistr setup, each federate has a connection to every other reldistr. With a large number of federates, there are more connections which must be polled in case a packet must be read and more connections to which a packet must be forwarded.

At the AAI and CT, the JTC federates were not explicitly using Data Distribution Management (DDM), so all reliable traffic was forwarded to all federates. All filtering for class and attribute–based subscriptions was done by the receiving LRC. This resulted in a tremendous amount of reliable traffic that was sent from each reldistr to every other reldistr. There may be other reasons for the performance boost when using the single reldistr, which may become apparent after further analysis.

2.3 DDM

DDM is a scheme by which federates can reduce the transmission and reception of irrelevant data through publication and subscription for object classes and attributes using regions in routing spaces [6]. Federates associate objects and attributes of object instances with regions in routing spaces. The RTI reflects updates to federates whose subscription regions, classes, and attributes match the region and class/attributes used by the publisher. In RTI1.3, the LRC uses DDM to route updates and interactions even if the federate does not explicitly use regions [7]. The LRC uses a default region in place of regions that the federate does not supply. Except for unique instances, all federates use a default region, so updates associated with default regions are routed to all federates; irrelevant updates are filtered by the receiving LRCs.

The effectiveness of DDM depends on the structure of the Federation Object Model (FOM) as well as the subscriptions of each federate. If DDM is used implicitly on a class/attribute subscription basis, then all filters can be specified directly in the FOM and RTI Initialization Data (RID); the federate, itself, is unchanged. DDM can also be used explicitly on an attribute subscription basis when the federate uses DDM API calls.

The FMT is an example of a JTC federate that might benefit from DDM. The FMT utilizes the standard Management Object Model (MOM) and MOM extensions to monitor and manage RTI

based federations. It is only interested in MOM data and has no interest in the more than 10,000 federate objects that may exist in the federation at any given time. Since the FMT is not subscribed to the object classes, it will not discover them; however, the RTI 1.3 LRC for the FMT obtains state for all of those objects if DDM is not used to filter them because the updates are associated with the default region. This results in unnecessary memory and network bandwidth usage. If the FMT is started while an exercise is already in progress, it can become heavily loaded during the initialization phase because the LRC is receiving the state for the irrelevant objects . It also causes the FMT to receive and discard irrelevant updates for the same objects. So far, the JIS development team has only done preliminary experiments with RTI 1.3's implementation of DDM on a class/attribute subscription. It is yet undetermined as to whether or not the JTC as a whole would benefit from the use of DDM because its use does have overhead implications.

After observing the performance in a large JTC exercise, it was apparent that the DOM's protocol for distributing object state did not scale well. This protocol was also effecting the performance of ownership attribute transfer operations. A modification to the RTI 1.3 DOM protocol was implemented that allowed the DOM to update changes more incrementally. Preliminary tests indicate that this protocol modification handles the heavy load much more effectively.

2.4 API Assumptions

During some of the JIS test events performed throughout the year (events which included both CBS and CSSTSS), the ADAPTOR was not providing CBS and CSSTSS with adequate functionality. CBS and CSSTSS perform an operation that requires that each federate send and receive messages in a very concise and dependent manner. API assumptions with regard to service calls were made during the development of the ADAPTOR that caused inconsistent behavior when CBS and CSSTSS attempted their operations. The ADAPTOR was using the *Register Object Instance* call, *Update Attribute Values* call, and *Unconditional Attribute Ownership Assumption* call in a synchronized manner with *Discover Object Instance* callback, *Reflect Attribute Values* callback, and *Request Attribute Ownership Assumption* callback. The RTI guarantees that each service call will provide the appropriate callback and/or return value. But does not guarantee that all services calls will behave in a FIFO manner (with regard to callbacks), and there is no interrelationship between all service calls.

3. Programmatic Lessons Learned

During the course of development, integration, and testing of the JIS, several lessons were learned with respect to the process that was employed.

3.1 Requirements Traceability

One of the first initiatives undertaken during development of the JIS was a systematic review and functional decomposition of the capabilities of the existing AIS. This set of functional requirements would form the basis of the JIS requirements. The requirements documentation was a key factor in establishing a test program that ensured that test steps were adequately developed that would exercise all of the required functionality in the JIS. Each test step was carefully mapped back to an existing functional requirement. The sequencing of test steps was also dictated by the criticality of the requirements. The documentation of requirements and the traceability to test steps also provided a means to track success over time, by test event, for each of the tested requirements. Besides providing a gauge of overall success, this also allowed us to identify areas in which functions that worked previously no longer worked. This provided the basic means of regression testing.

3.2 Integration and Testing

As cited in section 1.3, two major types of testing were performed with the JIS - internal MITRE lab testing and external site testing. Both of these were critical to the overall test strategy.

The primary benefit of testing within the MITRE lab is that it provided a stable environment for performing component testing of the RTI, ADAPTOR, and FMT. The basic functionality of each component could be tested as it was developed.

A number of support tools are used to emulate aspects of the JTC and provide diagnostic capabilities that could identify software problems. The test harness and test actor can connect to an ADAPTOR, replay log files, and generate ALSP messages. Both components are

a key factor in being able to replicate elements of the model interface without having to use a JTC model. Multiple test harnesses and test actors are often federated in such a way that the behavior of larger numbers of JTC models could be examined. Additionally, these tools provide the means to generate large amounts of message traffic without using models, which enabled us to perform load testing within the lab. The log management tool (LMT) provides the query and analysis of log files containing ALSP traffic both to and from multiple models and their ADAPTORs.

Throughout the summer and fall of 1998, copies of several JTC models were installed in the MITRE lab that extended realism of exercise sites. Copies of CBS, MTWS, AWSIM, RESA, the TACSIM ALSP Translator (TAT), and JQUAD provided the capability to test the majority of JTC interfaces. This had the added benefit of providing a better understanding of how the models really operated as well as uncovering differences between the actual interface and the documented interface. Maintaining these models within the lab permitted us to address integration issues continuously, without being limited to testing at an external site for a short period of time.

Although testing within the MITRE lab provided a realistic JTC environment, there were many problems that were not uncovered until the JIS was tested externally at user sites. Differences in network and communications architectures between sites forced us to address problems not immediately seen in the lab. As a result, much of the time spent in the field revolved around working through basic operational issues in running the software. Although this accounted for a significant portion of time, it was necessary to work through each problem as the JIS was intended to run on the different configurations supported at each site. In several cases, we allowed for an entire week prior to the actual test event just to install, configure, and initialize the JIS. During each event, differences in the physical configuration of each site were cataloged and used to develop a standardized set of operational procedures.

As much of the testing with the JIS was occurring in parallel to testing of RTI 1.3, on-site support with RTI personnel was a key factor in resolving problems. This allowed the ADAPTOR and FMT development teams to gain immediate feedback on RTI operations and also allowed the RTI development team to gain hands-on experience with JTC operations. It also significantly shortened the turnaround time for making code fixes to the RTI.

3.3 Operational Issues

One of the most important programmatic lessons that was learned during the JIS integration was the need to establish SOPs for running the JIS. This became apparent during many instances in which the lack of an SOP contributed to much greater downtime of the JIS than was necessary. Unfortunately, there is no other way to establish such procedures other than working through such operational issues during testing. The lessons learned from this experience are being documented in a JIS user manual that will carefully document startup, runtime, and recovery procedures for all aspects of running the JIS. These lessons will also be reviewed during planned JIS training classes at MITRE beginning in the fall of 1999, leading up to planned fielding of the JIS in the spring of 2000.

4. Current Status

Our first and foremost objective is to provide the JTC training audience with at least the same functionality, performance, and reliability as the current AIS. A secondary objective is to improve the usability of the interface, enable exercise controllers to diagnose problems faster, navigate control menus quicker, and provide automated federation data for technical after action reviews (AARs).

The functionality of the JIS, using RTI 1.3, currently meets the requirements of the AIS. All aspects of time management, object management, interest management, and exercise control (save, restore, communications) appear to have been successfully implemented and tested. Efforts over the next 6 months will focus on regression testing the functional capabilities in more stressful environments (large numbers federates, high load) and testing the functionality within RTI 1.3 Next Generation (NG).

Performance of the JIS appears to be improved from the CT, with the use of a single reliable distributor in the RTI, and improvements in the thread control of the ADAPTOR. Efforts in the near term will focus on benchmarking the combined performance of RTI 1.3 and the ADAPTOR under conditions similar to the largest JTC exercises - registering more than 10,000 objects and running 1:1 under periods of

peak load. Similar performance testing is planned for RTI 1.3NG

Reliability of the JIS is also much improved from the CT. Reliability goals continue to be the ability to run the JIS for a two week period continuously, with a minimum of failures and the ability to recover quickly from any problem. Improvements in the ADAPTOR interface since the CT now allow for more centralized control of ADAPTOR functions (those that cannot be controlled by the FMT). This has greatly reduced the time required to perform a restore operation. The FMT is undergoing several revisions - most notably a move from the existing Java Advanced Windowing Toolkit (AWT) graphics to the use of swing classes, which appears to provide much greater stability in the graphics interface.

In areas of usability, the FMT is implementing several new graphics screens that will provide the user with historical data on federation performance. This will allow tracking of load (object updates, deletes, and interactions to the infrastructure), federation rate, model availability, and downtime throughout the exercise. In the past, such metrics were often recorded manually and were cumbersome to summarize during technical AARs. The automated collection of this data in the FMT will improve the analysis of federation problems and will unburden exercise operations of much of their tedious job of data collection.

5. Future Efforts

The existing plans for testing of the JIS include a wide variety of test events at various locations throughout the fall and winter. With the intent to focus on resolution of operational issues, two additional exercise sites, the Korean Battle Simulation Center (KBSC) in Seoul, Korea and the Battlestaff Training School (BTS) in Hurlburt, FL, have been added to the list of test and demonstration events. Table 5.1 contains a complete list of these events.

As the JTC moves forward the emphasis for the JIS will shift from a focus on research and development activities to a focus on providing a stable configuration management baseline for the JTC user community. During the upcoming development cycle (FY00) the RTI baseline will be migrated to RTI version 1.3NG. This migration will provide the JTC users with infrastructure software that is composed of elements that are employed and supported by a much wider DoD base. The remaining developmental activities will revolve around efforts to improve the user interfaces and to leverage the existing lessons learned via a variety of product improvements.

Table 5.1 Major FY00 JIS Test and Demonstration Events

Date	Place	Test Event	Purpose
Aug 99	KBSC	JIS Demo	Demonstrate the JIS in the UFL test confederation
Sep 99	JTASC	JIS Testing	Perform testing with the JIS using the models planned for Unified Endeavor
Oct 99	BTS	JIS Testing / Demo	Replay portions of Blue Flag exercise using the JIS
Nov 99	JTASC	JIS Testing	Continued testing of JIS using same UE models
TBD	As Required	Functional Interface Integrations (FIIs)	Testing with FY00 versions of a subset of JTC models
TBD	JTASC	AAI	Technical test of the FY00 JTC
Mar 00	WPC	Confederation Test (CT)	Full JTC acceptance test

A wide range of activities will be performed to support the deployment of the JIS. Both on-site and formal classroom training on the ADAPTOR and FMT will be provided to supplement the currently available RTI training. The JIS development team will provide on-site support for the first exercise employing the JIS at each user site to ease the transition to the new product baseline. As the team moves through these efforts, an increased emphasis will be placed on capturing lessons learned. Two primary changes are projected at this point to the existing approach. The first will be the enhanced collection of data relative to both JIS and JTC performance through new capabilities contained within the ADAPTOR and FMT software. The second change will be through the collection of lessons learned inputs provided as direct feedback from the user community. In order to accomplish this the existing problem report tracking system will be enhanced to enable easier user access and improved reporting capabilities.

The JTC community will continue to document and analyze these lessons learned. Hopefully, as the community at large also migrates towards wide spread HLA implementation efforts such as ours to share the lessons learned will also increase. In conclusion, the ongoing efforts to

migrate the JTC will provide a wealth of lessons learned that should continue to be of benefit to a more diverse audience.

6. References

[1] Annette Wilson and Richard Weatherly, "The Aggregate Level Simulation Protocol: An Evolving System," Proceedings of the 1994 Winter Simulation Conference, Orlando, Florida, December 1994.

[2] Sean P. Griffin, Ernest H. Page, Zachary Furness, Mary C. Fischer, "Providing Uninterrupted Training to the Joint Training Confederation (JTC) During Transition to the High Level Architecture (HLA)," Proceedings of the 1997 Simulation Technology and Training (SimTecT) Conference, Canberra, Australia, 17-20 March, 1997.

[3] Dave Prochnow, Ernest H. Page, Bryan Youmans, "Development of a Federation Management Tool: Implications for HLA," 98S-SIW-103, 1998 Spring Simulation Interoperability Workshop.

[4] George McFadden, "Aggregate Level Simulation Protocol (ALSP) 1999 Joint Training Confederation Operational Specification," April 1999.

[5] Harry M. Wolfson, Steven B. Boswell, Daniel J. Van Hook, Stephen M. McGarry, "Reliable Multicast in the STOW RTI Prototype," 97S-SIW-119, 1997 Spring Simulation Interoperability Workshop.

[6] U.S. Department of Defense, High Level Architecture Interface Specification Version 1.3, 2 April 1998.

[7] Daniel J. Van Hook, James O. Calvin, "Data Distribution Management in RTI 1.3," 98S-SIW-206, 1998 Spring Simulation Interoperability Workshop.

Author Biographies

ZACH FURNESS is the Project Leader for the Aggregate Level Simulation Protocol (ALSP) project within the MITRE Corporation. Since joining MITRE in 1990, he has worked on numerous simulation programs involving the training, acquisition, and analysis of C4I systems. He holds a Master's Degree in Electrical Engineering and a Bachelor's Degree in Physics, both from Virginia Tech.

MARK ROBINSON is the lead software developer on the ALSP project. He has been supporting ALSP since joining MITRE in 1996. He holds a Bachelor's Degree in Computer Science from Western Michigan University.

CAROL CHIANG is a lead software developer for the RTI 1.3 and STOW RTI-s prototypes at MIT Lincoln Laboratory. Prior to joining MIT Lincoln Laboratory in 1994, she worked at Loral Advanced Distributed Simulation and BBN Systems and Technologies on SIMNET and DIS distributed simulation systems. She holds a B.S. in Electrical Engineering and an M. S. degree in Electrical Engineering and Computer Science, both from the Massachusetts Institute of Technology.

JOHN LOGSDON is the Project Director for the ALSP project within STRICOM. Since joining the government in 1982, he has worked as the PM/PD on projects ranging from the U.S. Navy's F-14D(R) & F-14A-B Upgrade projects to the U.S. Army's Corps Battle Simulation (CBS) and Warfighters Simulation (WARSIM) projects. He holds a Bachelor's Degree in Electrical Engineering from West Virginia Institute of Technology.

MITCHELL PRIMAS is a lead systems engineer at STRICOM where he is currently working as the lead project engineer for the ALSP program. He received the M.S. degree in Industrial Engineering from the University of Central Florida (UCF) in 1999 and the B.S. degree in Mechanical Engineering from the State University of New York (SUNY) at Buffalo in 1987. He has over 6 years of modeling and simulation systems engineering experience at STRICOM.