Modernizing Army Experimentation using OneSAF Objective System

Jennifer Lewis, Kirk E. Kemmler and Khoi Do
Science Applications International Corporation
Orlando, FL
jennifer.e.lewis@saic.com, kirk.e.kemmler@saic.com, khoi.m.do@saic.com

ABSTRACT

Training and Doctrine Command (TRADOC) is executing its plan to replace its primary entity driver in the Battle Lab Collaborative Simulation Environment (BLCSE). Replacing the existing multipurpose OneSAF Testbed Baseline (OTB) functionalities with OneSAF Objective System (OOS) will transition Army experimentation in the Advanced Concepts and Requirements domain to a fully capable environment for the study and testing of Future Combat Systems (FCS) capabilities. Because BLCSE maintains an aggressive analytical experimentation schedule, the transition from OTB to OOS must be completed in a short timeframe while preventing loss of functionality for remaining BLCSE federate applications. This paper discusses the technical issues associated with BLCSE's SAF replacement process, ranging from entity driver replacement to simulation message protocol adaptation. The paper specifically describes near-term activities associated with identification and resolution of interoperability issues and functionality gaps within a large-scale, highly-distributed simulation environment. In addition, the paper discusses potential enhancements to the BLCSE environment made possible by the integration of OOS, including behavior and modeling flexibility, varying entity fidelity and the introduction of OOS-based servers and tools.

ABOUT THE AUTHORS

Jennifer Lewis is a simulation engineer supporting TRADOC's Battle Lab Collaborative Simulation Environment. She holds a Master of Science degree in Computer Science with an emphasis in Telecommunications and Networking from the University of Texas at Dallas. She has designed and implemented network protocols for the telecommunications and defense industries for the past seven years.

Kirk E. Kemmler is a software engineer supporting TRADOC's Battle Lab Collaborative Simulation Environment.. He holds a Bachelor of Science degree in Computer Engineering from the University of Central Florida. Since January 2000 he has participated in the design and development of Man-In-The-Loop, Virtual, and Constructive simulation programs.

Khoi Do is a simulation software engineer supporting TRADOC's Battle Lab Collaborative Simulation Environment. He holds a Bachelor of Science degree in Computer Science from the University of Central Florida. He has developed and integrated military constructive and virtual simulations for the past eight years.

Modernizing Army Experimentation using OneSAF Objective System

Jennifer Lewis, Kirk E. Kemmler and Khoi Do
Science Applications International Corporation
Orlando, FL
jennifer.e.lewis@saic.com, kirk.e.kemmler@saic.com, khoi.m.do@saic.com

INTRODUCTION

This paper discusses the technical issues associated with modernizing Training and Doctrine Command's (TRADOC) Battle Lab Collaborative Simulation Environment (BLCSE), specifically transitioning its primary entity driver from OneSAF Testbed (OTB) to OneSAF Objective System (OOS). It provides background information about the federation and details design and implementation activities associated with the OTB-to-OOS transition. It also shows the results of BLCSE's technical integration events (TIEs) and BLCSE's OOS-centric path forward. The paper's intent is to familiarize the reader with BLCSE's technical environment and to provide solutions to potential interoperability issues other simulation environments may encounter as they transition to OOS.

TRADOC's Simulation Environment

BLCSE conducts analytical experimentation to support and give actionable recommendations for Doctrine, Organization, Training, Materiel, Leadership and Education, Personnel and Facilities (DOTMLPF) decisions. It is a persistent, large-scale simulation environment, geographically distributed among the Army's 18 Battle Labs and Centers. Twelve of these sites are proponents, or advocates, for major experimentation areas, such as maneuver support and missile defense, each with their own specific, and sometimes conflicting, simulation requirements.

As proponents, these sites do not merely host a federate in the federation. A proponent is expected to comprehensively represent its role in the experiment scenario. For example, as the dismounted infantry proponent, the Maneuver Battle Lab (MBL) at Ft. Benning, GA, may have up to 60 soldier simulations operating with one or more instantiations of OTB. In addition, the MBL will run numerous Advanced Simulation Technology, Inc. (ASTi) radio simulations through a communications effects server. Therefore, a single BLCSE site is often a complex federation in itself.

BLCSE is connected by a classified, virtual network hosted by the Defense Research and Engineering Network (DREN). A fully functional BLCSE federation involves more than 1,000 computers and network-addressable components. During a major experiment, the original BLCSE federation modeled up to 120,000 entities using Distributed Interactive Simulation (DIS) as its primary inter-federate communications protocol and using OTB as its primary federation entity driver.

Several technical issues arose from this original configuration. First, BLCSE's analysis capability requires consistent, repeatable results between scenario executions. By its best-effort, broadcast nature, DIS cannot guarantee such results, especially as entity count and site complexity increase.

Second, OTB's architecture requires a software engineer, rather than a user, to configure OTB for specific experiments. To complicate matters further, BLCSE's use of OTB has diverged from the official OTB v2.*x* baseline, and the community has developed several BLCSE-unique OTB variants, including several "server" applications that do not provide SAF-related functionality. BLCSE has faced interoperability issues as a result of lacking central configuration control for these variants.

For these reasons, TRADOC is in the process of transitioning BLCSE's inter-federate communications from DIS to the High Level Architecture (HLA) and transitioning its primary entity driver from OTB to OOS. These two transitions are separate engineering tasks, and this paper will speak to the OTB-to-OOS transition only. However, the paper necessarily refers to issues related to the DIS-to-HLA transition, and during the past year of continued effort, the team has found that the success of one transition often relies on the success of the other.

Interoperability with 3CE

The two separate transition tasks for BLCSE's modernization puts TRADOC into a position to participate in the Cross Command Collaboration Effort (3CE), a simulation environment combining: TRADOC's BLCSE; Research, Engineering and Development Command's (RDECOM) Modeling Architecture for Training, Research and Experimentation (MATREX); Army Test and Evaluation Command's (ATEC) Distributed Training Environment (DTE); and Program Manager Lead Systems Integrator's (PM LSI) Future Combat Systems (FCS) Simulation Environment. While the requirements of the 3CE federation are still evolving, the organizations have agreed to use the MATREX Run-time Infrastructure (RTI) under the HLA as the primary communication architecture. They have also agreed to use Objective Force OOS (OFOOS) v1.5 as the primary entity driver. As described above, BLCSE itself is a complicated federation-of-federations. As a part of 3CE, BLCSE will become part of an even larger puzzle. BLCSE's internal modernization effort will limit the interoperability issues that arise when piecing together the 3CE federation.

OOS TRANSITION REQUIREMENTS

The primary requirement for the transition from OTB to OOS is that the process put in place must prevent any loss of BLCSE functionality. This includes functionality provided by BLCSE-unique OTB variants, including the situational awareness (SA) server, the effects adjudication server, and the communications effects server. This primary requirement calls for a thorough understanding of how BLCSE uses its version of OTB and each of its OTB variants.

Secondly, the transition process must not adversely affect BLCSE's experimentation schedule. BLCSE normally conducts two to four major experiments per year, each of which requires approximately six months of preparation per site. In addition, the Battle Labs and Centers conduct multiple mini-experiments per year to prepare for or to augment the major experiments.

Finally, and perhaps most critical to the transition support team, is the short timeline to complete work. TRADOC wants BLCSE to be fully transitioned to OOS by the time OOS v1.5 is released on October 1, 2007. To meet this deadline, the transition team is executing a series of five TIEs from February to October 2007. These events have stressed the entire BLCSE federation, from OOS to the RTI to the DREN connecting the sites, in a way never done before. The next sections will describe, in detail, the activities performed in preparation for the TIEs as well as notable observations and solutions from past TIEs.

ENVIRONMENT ANALYSIS

To prepare for the OTB-to-OOS transition, the transition team compared OOS capabilities to the ways BLCSE is currently using OTB. Although many issues were considered, two areas warranted the most attention: gaps in modeled entities and differences in SA architecture.

Model Gap Analysis

The team performed an entity and unit comparison between Objective Force OOS version 1.0 and the OTB versions used for FCS v1 and at the Unit of Action Maneuver Battle Lab (UAMBL). Fundamental differences in simulation architecture prevented a straight forward one-to-one comparison. Within OTB much of the entity and unit capabilities lie within OTB Task Frames, which are equivalent to orderable behaviors in OOS. This is not necessarily so in OOS, where entity and unit capabilities are composed with selectable behavioral and physical components. The OOS-composed components define behaviors such as movement, sensing, vulnerability and communications.

The only area that lends itself to a direct comparative analysis is the graphical user interface (GUI) capabilities of each simulation. The team identified and compared 30 high value GUI features, and they found that 28 of the features were represented in OOS. OOS does not implement two of the features directly, but the team found indirect ways of representing the functionality.

The team completed compared OTB and OOS entities by entity name comparison, using the OTB entity models file and the OOS medium resolution entity compositions. The OTB entity models file functionally defines OTB entities in the entity name. Based on the entity name, the team matched OTB entities with OOS entities. In a comparison of only United States and Soviet entities, the team found that OOS lacked about 30 percent of the entities that OTB contained. In addition, OTB contains many country specific entities that OOS does not directly represent, such as forces from Canada, Czech Republic, Germany, and Slovakia. The BLCSE federation does not typically use these country-specific entities, so they were given little comparison weight during the entity analysis.

The team compared OTB and OOS units by unit name comparison, using the OTB unit models file and OOS medium resolution unit compositions. The majority of missing units in OOS were foreign country units and aviation units.

The team compared the OTB Task Frames and OOS Composite Behaviors using the OTB provided Task Frame user documentation and the behavior descriptions available from the OOS Behavior Composer Tool. The descriptions contained in the OTB documentation provided a functionality assessment the team used to compare like functionality in OOS. Because of the transition team's previous experience with OOS, the Task Frame descriptions provided enough of a functionality description to match OOS behaviors.

A straight forward behavior-to-behavior comparison between OTB and OOS revealed that OOS lacks about 16 percent of the orderable behavior functionality that OTB contains. However, with further investigation, the team discovered that two-thirds of the missing orderable behaviors are represented in OOS indirectly, through methods such as agents and direct intervention. Furthermore, by making allowances for fundamental architectural differences between OTB and OOS, the team found that only eight percent of the missing behaviors could not be represented in OOS without software developer intervention. That is, 99 percent of OTB functionality is represented in OOS. The missing one percent of behavior functionality in OOS is in the area of vehicular part articulation and manipulation of construction or mine attachments, e.g. plows and sensor arms. OOS does not model or perform this functionality as an explicit behavior; rather, it is implied. Most behaviors which utilize these parts model only the time delays associated with their use instead of physically modeling their use.

To conclude the OTB and OOS comparison, the transition team compared physical models, using lists obtained from the applications' source libraries and source code documentation. The team documented and compared 93 OTB physical models to OOS physical models. They discovered that, on a direct comparison, OOS lacked approximately 30 percent of the models OTB contains. Of the missing 30 percent, the team found that 22 percent are soon-to-be released OOS naval models. The team found that another 14 percent are modeled functionality provided by other means within OOS behavior or agent capabilities. That is, OOS completely lacks approximately 20 percent OTB's physical models. The missing physical models are naval based systems, radar systems such as Semi-active Radar (SAR) and Laser Radar (LADAR), and models in support of part and component articulation. The team reported the results of the gap analysis to the BLCSE community, who are now reviewing the need for BLCSE development in these areas.

Situational Awareness Analysis

As BLCSE transitions from OTB to OOS, from DIS to HLA, and from a standalone federation to a member of the 3CE, there are certain external, OTB-based servers that will no longer be supported or have become redundant. One of the external servers that will be diluted into the environment is the SA Server.

BLCSE's SA architecture generates network data that describes the perceived battlefield. The SA Server receives ground truth data from the federation, performs target fusion, and then distributes the perceived data to all interested federates. The SA Server determines which federates are interested by maintaining a database of entity locations and assigning each entity to a dynamic SA cell. The perceived data includes the Current Operational Picture (COP) and a mechanism for initiating and tracking indirect-fire missions. The Mobile Command and Control (MC2) federate operates via the COP, and therefore, it relies on the BLCSE SA Server to forward properly translated salute reports and perceived entity messages. Upon receiving data from the SA Server, MC2 processes the salute report messages and disperses them back out as fire mission target messages.

The team anticipates replacing BLCSE's current SA architecture using a combination of the MATREX Command, Control and Communications (C3) Grid and newly-released modifications to OOS's Data Distribution Management (DDM) implementation. The C3Grid federate will replace the target fusion and distribution capabilities of the current SA Server. Using the C3Grid will allow BLCSE to interoperate more efficiently with the 3CE federation, since many 3CE federates already use the C3Grid. In addition, OOS is now capable of defining interaction regions from a data file, subscribing to interactions with a region and sending interactions for a specifically defined region. This OOS functionality provides C3Grid the capability to send and receive interactions to and from OOS federates based on subscription regions, removing all need for the complex SA cell structure currently in place. OOS salute reports will now be translated to an HLA interaction, sent, and then received by command and control systems like MC2 without using the existing SA Server.

IMPLEMENTING OOS INTEROPERABILITY

OOS provides the user with ready-to-use composite behaviors. These behaviors are mostly used in, and currently designed for use in, either standalone mode or distributed mode. Some of these behaviors do not have the capability to operate effectively in interoperability mode. If the behavior requires interaction between two entities that are not modeled on the same federate, a software engineer must make changes to the OOS interoperability packages to allow the interaction. An engineer must also implement a method, such as a listener, an agent, or a user-initiated behavior, to provide the needed Runtime Data Model (RDM) object data.

HLA Interoperability

In order for OOS to communicate in HLA, OOS internal objects and interactions need to be translated to their Federation Object Model (FOM) equivalents. These are objects and interactions such as entities states, firing and detonation events. There is one main eXtended Marking Language (XML) document called "mappings.xml" that identifies how to map between OOS and HLA. The mapping process involves mapping each attribute in the HLA interaction or object that interests OOS and the FOM requires. Each attribute can be subscribed to, published or both. A Java class translates an OOS data type to an HLA data type and vice versa. These Java classes are referenced by the "mappings.xml" during the mapping process.

There are additional XML mapping files that are needed to translate OOS to FOM enumerations and vice versa. These handle enumerations such as entity types and munitions types. With this architecture, OOS can support multiple FOMs without changing its internal objects and interactions.

Because of the simplicity of this approach, OOS is limited to using FOM platform enumerations that may not provide enough values to cover all of the OOS entity variants. To prevent inaccurate mappings, the team created a new FOM attribute, which will contain the name of the OOS entity composition. Instead of using mappings, the receiving OOS will convert this attribute to allow the exact entity to be displayed.

OOS version 1.0 was released with numerous MATREX FOM objects and interactions mapped. However, in order for OOS to replace OTB in BLCSE without losing critical functionality, additional interactions are required. Most important of these are Call For Fire, Asset, and Salute Report.

Call For Fire Interaction Mapping

BLCSE was posed with a need to issue a fire order from an OOS federate to the Fire Support Simulation (FireSim) federate without using Command, Control, Communications, Computers and Intelligence (C4I) capabilities. To accomplish this, the transition team implemented a user-initiated behavior to provide the data needed to fill in the FOM class parameters. The needed data is provided in the form of an RDM Call For Fire message that is filtered by interoperability classes to determine if the message recipient is a recipient external to the issuing federate. If the message recipient is an entity that is modeled on an external federate, then the converter translates the RDM message data to a FOM message data. The team added a Java converter class to translate the OOS RDM Call For Fire class objects into an appropriately formatted message based on the MATREX FOM. Once all of the OOS Call For Fire data is translated into the FOM object structure, the message is sent from OOS. Systems such as FireSim can receive and translate these messages from OOS into an actionable engagement on a target or location specified within the Call For Fire interaction.

Asset Object Mapping

BLCSE federates use the Asset object to learn about external firing assets. They use this information to order the correct asset to carry out a Call For Fire mission. In order for OOS to receive asset information and use it to issue a Call For Fire order to a BLCSE federate, the team created an Asset converter Java class. As input, the class uses the ID of the platform for which the Asset object is reporting, munition type, authorized amount, and current amount. Upon receiving an Asset object, OOS finds the external entity for which the Asset object is reporting and updates its basic load supply information. The OOS user-interface is also updated with the new information. Using the information provided by the Asset object, the user can input correct munition type and quantity to issue the Call For Fire order to a BLCSE federate.

Salute Report Interaction Mapping

BLCSE federates use Salute Report interactions to share perceived truth. As in the previous examples, the team created Java classes to convert the parameters from OOS internal to the FOM format, allowing OOS to distribute Salute Report messages to external federates. Currently, OOS has the capability to distribute Salute Reports to external federates but does not make use of Salute Report interactions generated from external federates. The OOS messaging framework requires messages to contain a recipient and does not currently provide the SA architecture or COP BLCSE requires. However, the team is actively working in this area, in coordination with OOS architects and those in the community involved with SA and the COP.

INTEROPERABILITY WITH DIS FEDERATES

BLCSE is composed of a tightly woven set of DIS-native simulations that have interoperated with one another for approximately six years. Over the past two years, as part of the DIS-to-HLA transition, these simulations have begun using specifically-designed Middleware software to allow the simulations to "speak" HLA. The integration of a new simulation, in this case OOS, brings new interoperability issues. One of these issues is that OOS did not display firing lines for fire events generated by BLCSE Middleware federates. The detonation interaction in the FOM has numerous optional parameters. OOS requires two of the optional parameters to properly display firing lines. To resolve the problem, the team made minor changes to Middleware to allow existing BLCSE federates to fill in the necessary optional attributes.

Another issue involves BLCSE federates that join the federation after OOS federates. These federates do not see entities that belong to OOS federates. The BLCSE federation had never encountered this issue before because previously all federates sent entity heart-beat messages at set intervals which contain entity state information. As is common in HLA-native simulations, OOS does not send regularly-scheduled entity heart-beat messages. Instead, OOS only sends entity status information if there is a change to report. To resolve this issue, upon joining the federation, all BLCSE Middleware federates request attribute updates for all RTI objects in which they are interested. The RTI then triggers existing OOS federates to resend their entity states, regardless of whether those states have changed. In a future version, OOS will implement regularly scheduled heart-beat messages.

The team also encountered a problem where OOS and BLCSE Middleware federates could not discover each other's aircraft. Entity representation in the MATREX FOM follows a parent-child hierarchical relationship. BLCSE Middleware federates model air platforms as a parent "platform" object. OOS, on the other hand, models them as an "aircraft" object, which is a child class of "platform". The team solved this problem by modifying the Middleware to publish and subscribe air platforms as "aircraft" objects.

TECHNICAL INTEGRATION EXECUTION

During 2007, the transition team led a series of TIEs to test the work described above. As of this writing, only two of six planned TIEs have been executed. During TIE2, the team organized a controlled federation of seven geographically-dispersed sites using the MATREX RTI v5.0. The federation maintained a stable state that continued to operate satisfactorily for approximately eight hours with a high entity count of 3,867 entities. Approximately 150 of these entities were high resolution entities modeled by Middleware-integrated BLCSE federates, including FireSim, the Extended Air Defense Simulation (EADSIM) and OTB. The remaining entities were modeled by OOS as medium resolution entities. Figure 1 shows a deliberate increase of distributed objects in the TIE2 federation as federates performed certain actions specified by the battlemaster, such as joining the federation and running scenarios. At the 23rd federate action, federates began adding entities to the federation at once. The team then noted a spike in RTI objects, when neither entity nor federate increase seems to justify the RTI increase.

Before achieving this point, however, TIE1 and TIE2 encountered many technical issues associated with running OOS in a large-scale, heterogeneous federation such as BLCSE. The following subsections describe the symptoms of and solutions to the problems encountered.

Using OOS on the DREN

One of the issues the team encountered while using OOS over the DREN is receiving HLA object updates before receiving their object discovery callbacks. This issue causes OOS to generate Java exceptions. OOS expects to receive object discoveries before receiving object updates. Enabling the Lazy Object Discovery feature within MATREX RTI Initialization Data (RID) file fixed this problem. The MATREX RTI v5.2, released in May 2007, has Lazy Object Discovery enabled in software, eliminating the need for the RID file parameter.

Another issue the team encountered is that not all OOS systems at remote sites see the same number of entities being modeled in the federation. Entity states in the

MATREX FOM are always sent using User Datagram Protocol (UDP) over multicast addresses. Therefore, if a site's network infrastructure blocks certain multicast addresses, that site may not be able to see all of the entities modeled in the federation. To determine if this is the problem, the transition team will modify the RID file to send everything as Transmission Control Protocol (TCP), which does not use multicast addresses, regardless of what the FOM specifies. If sending entity states as TCP solves the problem, the team will involve network design engineers to ensure the network infrastructure is not blocking critical multicast addresses. Since UDP packets are sent best effort, drops in network packages may also be part of the cause. Lowering the network's Maximum Transmission Unit (MTU) size may also help solve the problem.

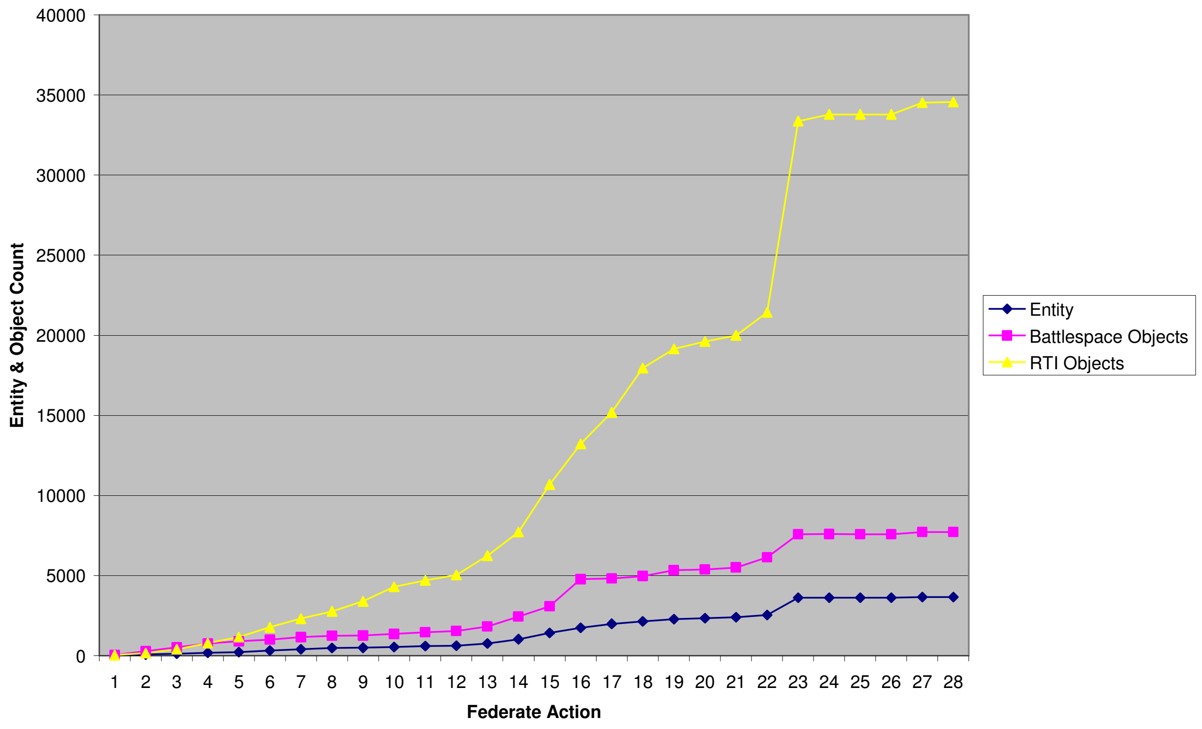

Figure 1. Controlled Federate Actions Result in a Steady Increase of Entity and RTI Objects

Combining Multiple OOS Compositions

When running OOS with combined composition functionality on the same personal computer (PC), the team found the federation tends to become unstable and susceptible to federate runtime failure. Insufficient hardware resources were the cause of most federate crashes. Combining the functionality of the Management and Control Tool (MCT), entity modeling (Simcore), and interoperability compositions overwhelms typical hardware capabilities, even when modeling only the number of entities recommended by OOS.

To alleviate the problem, the team recommends running OOS in a clustered environment, distributing computing resources among multiple computers. A recommended cluster configuration requires a single PC running as the HLA interoperability manager, two to three PCs running as the SimCore, and as many MCTs as needed for user interaction. The MCT currently requires the most computing resources and is the limiting factor in any resource configuration used to operate an OOS federate. The MCT handles all GUI functionalities and environment databases. During experiment execution, the hardware resource that usually causes the slowness and instability of an OOS federate is the Random Access Memory (RAM).

Figure 2 shows the number of entities an OOS system was able to model in relation to the amount of RAM available to the system. The x axis labels give the site and amount of RAM available at that site, and the y axis lists entity count. Notice that Ft. Sill was able to model almost 200 more entities than any other site, even with lower hardware capabilities than other sites. During

TIE2, Ft. Sill had two machines each with four gigabytes of RAM. The reason Ft. Sill was able to remain stable while modeling so many entities on the network is because Ft. Sill was not discovering all of the external entities in the federation. Ft. Sill discovered approximately 63 percent of the total entity count of the federation. This low entity discovery rate provided Ft. Sill's systems with the resources to model a higher entity count, while maintaining a stable interface to the federation.

Running on Unsupported Operating Systems

OOS officially supports Windows XP and Red Hat Enterprise 4. However, the team had issues running OOS in HLA interoperability mode in a Windows XP environment. The problem is that certain MATREX RTI installations did not include the JavaBinding.jar file that OOS requires. Once the team installed a correct Java binding file, such as the jar file with the MATREX RTI version 5.0, OOS ran as expected on Windows XP.

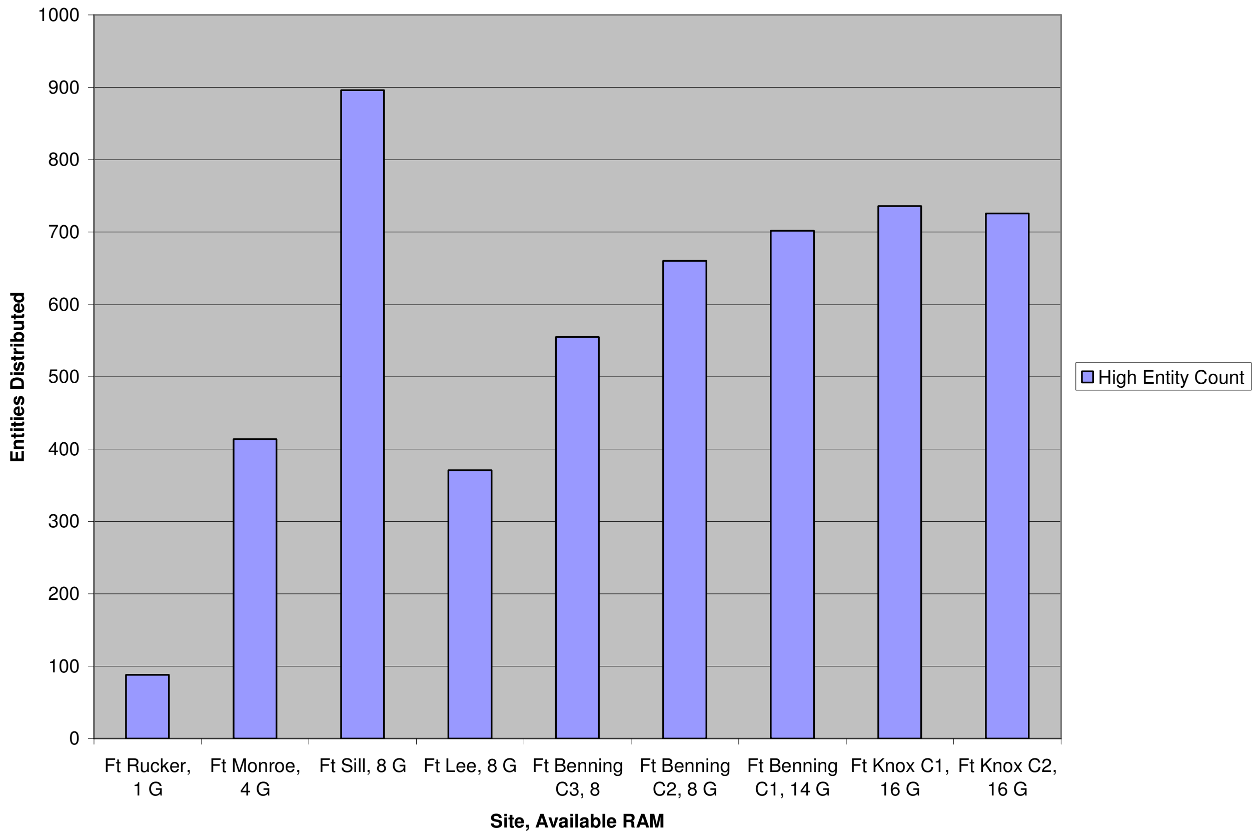

Figure 2. Available Computer RAM vs. Maximum OOS-Medium Resolution Entity Count

In addition, TRADOC wants to use an open source Linux flavor. In this case, the transition team recommends using CentOS 4.4 since it is 100 percent binary compatible with the officially-supported Red Hat Enterprise 4. However, the team also successfully ran OOS on other Linux operating systems, such as Fedora Core 5 and 6. Again, the team installed the correct Java package, since the Java packages included with those operating systems are not OOS-compatible. Also, the team discovered that OOS will not compile on Fedora Core using the officially-supported gcc version 3.3.2. However, the team compiled successfully using gcc version 3.2.3.

Using the Mak RTI

Currently, BLCSE uses the MATREX RTI but plans to use the Mak RTI in the future, mainly for its support for IEEE 1516 standard. However, OOS provides a MATREX RTI interface, and the team discovered some issues when using the Mak RTI. First, the Mak RTI Java binding package is not compatible with OOS. The class names and package path are different from those of MATREX RTI Java packages. However, the team made some simple RTI configuration changes, which allows OOS to be compile and runtime compatible with the Mak RTI using the same version of the Java binding jar file that is used by the MATREX RTI.

The second issue arises from using the MATREX RTI Java binding jar file, as described above. When a federate sends an updateAttributeValues callback to the RTI, the "tag" parameter causes null pointer exceptions. The MATREX RTI resolves the allowed NULL "tag" parameter in updateAttributeValues calls to an empty string, whereas the Mak RTI leaves the "tag" attribute as is. BLCSE Middleware federates always set the "tag" attribute to NULL. This causes the MATREX Java bindings to receive an unexpected NULL value, resulting in a NULL pointer exception in OOS. The latest MATREX release for RTI v5.2 updates the Java bindings to check for a NULL "tag" value.

FUTURE OPPORTUNITIES

BLCSE's OTB-to-OOS transition provides other opportunities for modernization besides simply replacing OTB with OOS. The following sections give a glimpse of BLCSE's OOS-centric path forward.

Development of OOS-based Tools

As aforementioned, BLCSE is currently using various OTB-based tools to handle functionalities such as situational awareness, damage effects, communication effects, and area of interest. OOS is a highly composable system, capable of replacing BLCSE's existing OTB-based tools. Different tools, or capabilities, can be added into or removed from an operating OOS simulation with relative ease, allowing that OOS simulation to fit the role requested by the federation. Tool development can follow an approach similar to the one described for OOS's HLA interop module. The tools listen to HLA messages, process them, and forward the results to the appropriate federate. The goal for these tools is to remove the processing burden from simulation federates. The specialized tool federates handle the responsibility, then hand off the processed information to the simulation federates for consumption. These tools can leverage Data Distribution Management (DDM) to optimize network traffic and further reduce the processing load on simulation federates. Although more analysis work is needed in this area, BLCSE's ultimate goal is to be completely OOS-based.

Replacement of Other BLCSE Models

Currently, OOS has many low fidelity and medium fidelity physical and behavioral models. These adequately perform modeling needs for a constructive simulation. In BLCSE, most of the federates perform high fidelity modeling, such as EADSIM for aircraft modeling and FireSim for fires modeling. In the past, OOS has been used in a similar way to implement a high-fidelity gunnery trainer known as the Common Gunnery Architecture (CGA). Due to the highly composable nature of OOS, it is possible to have a mix of fidelity models running at the same time. As in CGA, the main entity is composed of all high fidelity models such as sensor sights, turret system, stabilization system, laser designators, computer targeting system, and munition flyout. On the other hand, the target entities in the simulation are low and medium fidelity models. In order to replace existing BLCSE models, it is critical to obtain the existing models' algorithm, modeling data and, if possible, source code. It is cheaper and faster to create models within OOS, taking advantage of model reuse, rather than going through the physical knowledge and design phases of model development. Improvements can be made during this process with new data and algorithm refinements.

The transition team recognizes that it may not be cost-effective to create OOS models to replace existing models that are sufficiently complex or detailed. In this case, the team anticipates implementing ownership management handoffs using the HLA. While all entities would be created by OOS, the system would give ownership of entities over to the higher fidelity simulation when required. For example, assume an OOS-created M1A1 fired at a target. Using the RTI service to request attribute ownership, OOS would request FireSim to "own" the M829A1 munition entity, allowing FireSim to perform the detailed physical modeling for the munition flyout. OOS might request this service because it is aware that FireSim can model this flyout more precisely. OOS might also request this service to reduce its own modeling load. Technically, implementing ownership management is a trivial problem. However, the design of each federate application, specifically its ability to create new entities on-the-fly, must be thoroughly assessed before implementation begins.

SUMMARY

The size, complexity and schedule of the BLCSE federation present a number of challenges to its OTB-to-OOS transition. BLCSE's transition is stressing the functionality and performance of OOS's interoperability mode to its limits. This unparalleled stress test revealed many technical issues to be resolved in areas ranging from OOS and Middleware source code, RTI software, network design and workstation configuration. The transition team has found solutions for many of these interoperability issues. However, some areas such as SA will require a significant amount of additional work and coordination with other organizations in the community.

Although BLCSE intends to use OOS as its primary entity driver beginning in October 2007, removing all vestiges of the OTB-centric federation will require more time and effort. The transition team will continue to add complexity to TIE scenarios to further identify and resolve OOS interoperability issues. In addition, the composable nature of OOS provides BLCSE an array of options to further modernize Army experimentation.

ACKNOWLEDGEMENTS

The authors acknowledge the support of the following colleagues while writing this paper: Karyn Eusey, Paul Monday, Lawrence Rieger, Paul Hanover and Alex Jones.

REFERENCES

Hanover, Paul; Progress Report on the Battle Lab Collaborative Simulation Environment; *Proceedings of the 2004 Interservice/Industry Training, Simulation and Education Conference.*

Rieger, Lawrence and Lewis, Jennifer; Integrated Middleware for Flexible DIS and HLA Interoperability; *Proceedings of the 2006 Interservice/Industry Training, Simulation and Education Conference.*