Advances in Gateway Products and Processes for LVC Simulation Environments

Robert Lutz, David Drake, Nathaniel Horner
The Johns Hopkins University
Applied Physics Laboratory
Laurel, MD
Robert.Lutz@jhuapl.edu, David.Drake@jhuapl.edu, Nathaniel.Horner@jhuapl.edu

Michael O'Connor	**Kurt Lessmann**	**Dannie Cutts**
Trideum Corporation	**Trideum Corporation**	**AEgis Technologies**
Huntsville, AL	**Huntsville, AL**	**Huntsville, AL**
Moconnor@trideum.com	**Klessman@trideum.com**	**Dcutts@aegistg.com**

ABSTRACT

Integration of Live, Virtual, and Constructive (LVC) Modeling and Simulation (M&S) assets within a single, unified distributed simulation environment is commonplace today as a means to address the diverse needs of the training and test communities. Such environments frequently transcend the use of a single simulation architecture or Simulation Data Exchange Model (SDEM), as the organizations that participate in distributed LVC events generally want to employ the local standards and conventions with which they are most familiar. Different protocols and data formats are reconciled in modern development approaches through the use of gateways. Gateways provide a variety of translation and other services in LVC events, enabling operation across dissimilar architectures.

Despite the many success stories associated with gateway use in LVC events, there are several well-documented issues with gateways. Most of these are related to a general lack of supporting products to allow users to discover the gateway capabilities they need, along with a corresponding lack of products and standards for gateway configuration. These problems have resulted in a proliferation of dissimilar gateways that provide redundant capabilities, but which must be managed and maintained separately. The need to address these inherent inefficiencies has led to Department of Defense (DoD)-sponsored efforts to reduce the costs and to increase the effectiveness of gateway utilization in user programs.

This paper will describe the products developed under the Live-Virtual-Constructive Architecture Roadmap Implementation (LVCAR-I) effort to address the core goals in the gateway area. Examples of such products include Extensible Markup Language (XML)-based formal languages to support gateway selection and configuration, and specification of standard benchmarks for characterizing the performance of gateways. These products form the foundation for a suite of automated tools and supporting processes that will strongly facilitate gateway discovery and configuration in the future.

ABOUT THE AUTHORS

Robert Lutz is a principal staff scientist at the Johns Hopkins University Applied Physics Laboratory (JHU/APL) in Laurel, MD. He has over 30 years of experience in the design, implementation, and evaluation of M&S systems for military customers. Currently, he is leading the LVCAR-I "Systems Engineering (SE) Process" and "Common Gateways and Bridges" tasks, and serves as the U.S. Navy's Broad Area Maritime Surveillance (BAMS) Program M&S lead in the airspace integration area. He also leads several M&S standards initiatives within the Simulation Interoperability Standards Organization (SISO), including the Object Model Template (OMT) component to the Institute of Electrical and Electronics Engineers (IEEE) 1516 High Level Architecture (HLA) standard, and the IEEE 1730 Distributed Simulation Engineering and Execution Process (DSEEP) standard. He also serves as a regular guest lecturer in The Johns Hopkins University (JHU) Whiting School of Engineering.

David Drake is a member of the Senior Technical Staff in the Global Engagement Department at JHU/APL, performing research and development in the area of modeling and simulation supporting the U.S. government and military. Mr. Drake is currently the lead on the LVC "Service-Oriented Architecture" task and supports the LVCAR-I "Common Gateways and Bridges" task. Mr. Drake has 9 years experience in M&S design, development, and standards and 23 years as a computer security professional in computer security design, implementation and evaluation. Mr. Drake is a member of the SISO Standards Activity Committee (SAC) and is the secretary of the SISO Board of Directors (BoD). Mr. Drake received his bachelor's degree in Mathematics from SUNY at Buffalo. He is published in the areas of simulation, service-oriented architecture, grid computing, and security, and has a patent on the process for enterprise-wide intrusion detection.

Nathaniel Horner is an Associate Professional Staff member in the Aerospace Systems Design Group at JHU/APL, where he has served as Project Manager and Technical Lead on several modeling, simulation, and analysis (MS&A) efforts for U.S. Air Force clients. He is also heavily involved in projects using conceptual modeling and systems engineering methods to improve the efficiency and effectiveness of simulation development and usage. Mr. Horner holds master's degrees in Computer Science and Systems Engineering from JHU and the University of Virginia, respectively, and is a member of the JHU systems engineering faculty.

Michael O'Connor is a Senior Principal Engineer at Trideum Corporation. Mr. O'Connor has more than 20 years experience in M&S. He has been a key participant in the development of distributed modeling and simulation standards, including IEEE 1278 and IEEE 1516. He has held many positions in the community, including Chairman of the SISO Standards Activities Committee and Vice-Chairman of the SISO Executive Committee. Mr. O'Connor previously led the development of ITT's Chemical, Biological, Radiological, and Nuclear Simulation Suite, which supports Distributed Interactive Simulation (DIS), High Level Architecture (HLA), and Test and Training Enabling Architecture (TENA). He holds a bachelor's degree in Computer Engineering from Auburn University, and a master of science in Computer Science from the University of Alabama in Huntsville.

Kurt Lessmann is a Vice President at Trideum Corporation, specializing in providing solutions and services in the area of test and evaluation, M&S, systems analysis, and information technology. Mr. Lessmann has over 15 years experience in supporting LVC integration activities for large-scale Joint Distributed Test activities. Mr. Lessmann also supports the test community by providing Object Model design, software and range system design and integration, and application of M&S techniques. He currently supports the TENA project and the Joint Mission Environment Test Capability (JMETC) Program Office as a senior Event Lead. Mr. Lessmann holds a bachelor's degree in Aerospace Engineering from Auburn University.

Dannie Cutts is a Senior Computer Scientist with AEgis Technologies Group Inc., supporting the United States Joint Forces Command (USJFCOM) Joint Advanced Training Technologies Laboratory (JATTL) in Suffolk, VA. He has over 20 years experience in M&S for NASA and DoD and has been involved with the HLA since 1995, serving on the Interface Specification and Time Management Working Groups. He has provided HLA Training and Cadre support for the Defense Modeling and Simulation Office (DMSO), and currently serves on the IEEE Drafting Group for the HLA IEEE 1516 standard. Mr. Cutts is a Certified Modeling and Simulation Professional (CMSP). At USJFCOM he is involved in efforts to improve interoperability between LVC assets for Joint Training.

Advances in Gateway Products and Processes for LVC Simulation Environments

Robert Lutz, David Drake, Nathaniel Horner
The Johns Hopkins University
Applied Physics Laboratory
Laurel, MD
Robert.Lutz@jhuapl.edu, David.Drake@jhuapl.edu, Nathaniel.Horner@jhuapl.edu

Michael O'Connor
Trideum Corporation
Huntsville, AL
Moconnor@trideum.com

Kurt Lessmann
Trideum Corporation
Huntsville, AL
Klessman@trideum.com

Dannie Cutts
AEgis Technologies
Huntsville, AL
Dcutts@aegistg.com

BACKGROUND

Simulation is a critical enabler of many different functions in modern acquisition programs. While individual standalone models and simulations are the tool of choice for many systems engineering (SE) activities, the use of *distributed* simulation has increased sharply in recent years. Distributed simulation leverages high-speed networks and supporting simulation services to link together multiple existing modeling and simulation (M&S) assets into a single unified simulation environment. Test and Evaluation (T&E) and training are generally considered to be the most frequent users of this technique, as combinations of live, virtual and constructive M&S assets (including hardware-in-the-loop) are frequently required to support program requirements.

Several different simulation architectures are currently in active use within the U.S. Department of Defense (DoD), including its coalition partners. Examples of such architectures include Institute of Electrical and Electronics Engineers, Inc. (IEEE) 1278.1-1995 Distributed Interactive Simulation (DIS), the Test and Training Enabling Architecture (TENA), and the IEEE 1516 High Level Architecture (HLA). While all of these architectures provide services that support runtime interoperability among disparate simulation systems and supporting utilities (e.g., viewers, loggers), each individual architecture has been optimized for a particular class of user. For instance, DIS has been designed for real-time applications at a platform level of representation, while TENA is mainly focused on interfacing range assets, command and control (C2) systems, and simulations in support of T&E applications. Users of these architectures have coalesced over the years into established user communities, where tools and standard practices for developing and employing distributed simulation environments using the architecture are commonplace.

In some situations, sponsor requirements may necessitate the selection of simulations whose external interfaces align with different simulation architectures. This is known as a *multi-architecture simulation environment*. There are many examples of such environments within the DoD, especially in areas that require broad participation across disparate user communities (e.g., joint training and experimentation). When more than one simulation architecture must be used in the same environment, interoperability problems are compounded by the architectural differences. For instance, middleware incompatibilities, dissimilar metamodels for data exchange, and differences in the nature of the services that are provided by the architectures must all be reconciled for such environments to operate properly. This not only raises additional technical risk but, in addition, the additional resource consumption necessary to adjudicate these architectural differences affects cost and schedule risk[1].

Because of perceived increases in the number of multi-architecture simulation events anticipated in the future, along with the associated increase in costs, DoD sponsored an initiative to examine the differences among the major simulation architectures from a technical, business, and standards perspective, and to develop a time-phased set of actions to improve interoperability within multi-architecture simulation environments in the future. This initiative is called the Live-Virtual-Constructive Architecture Roadmap (LVCAR). The development of the Roadmap began in

the spring of 2007 and continued for approximately sixteen months. The result of this activity was a final report and supporting documentation that collectively totaled over a thousand pages[2].

One of the key recommendations from the LVCAR Final Report focused on improvements to gateway and bridge capabilities. The term "bridge" in this context refers to intelligent translators that link together enclaves of simulations that use the same underlying simulation architecture, such as a bridge between two separate simulation environments that both use HLA. A "gateway" is also an intelligent translator, but it is designed to link simulation enclaves that use dissimilar architectures, such as a gateway between simulations that use TENA as its external interface on one side of the translator and DIS on the other. Since LVCAR continues to focus on multi-architecture interoperability, LVCAR recommendations mainly emphasized improvements to gateways rather than bridges. Note that one of the key functions of a gateway is to adjudicate the differences in the format, syntax, and content of the data that is exchanged at runtime among cooperating simulations. Known as an "object model" in some communities, this data representation is referred to more generically as a Simulation Data Exchange Model (SDEM).

LVCAR implementation (LVCAR-I) began in 2009. Early efforts focused on the characterization of the state of the gateway marketplace[3] and on evaluating potential strategies for addressing deficiencies and problems identified by gateway users[4]. These "challenges", and an associated set of gateway requirements, are discussed in the next two sections. The strategy that addresses these challenges focuses on the development of new products that improve how users select, configure, and employ existing gateways. This approach has the advantage of facilitating user productivity gains in the short-term at relatively low cost, while lowering the costs associated with migrating to more advanced gateway solutions in the future. A description of these products is provided later in this paper.

GATEWAY CHALLENGES

Gateways are a key enabler of multi-architecture simulation environments. Although there are many success stories with respect to the use of gateways on multi-architecture developments, there have also been some reported problems. Since there is no such thing as a "common" gateway across (or sometimes, even within) user communities, managers of some LVC environments are often unaware of reuse opportunities for needed gateway capabilities. Thus, from a historical perspective, many programs have built their own gateways from scratch based on their immediate needs, with little or no attention paid to potential reuse. This has led to an unnecessarily large number of gateways in the LVC community, many of which are ad hoc, have little documentation, and have no visibility outside the projects for which they were designed. This is, of course, very inefficient from a DoD enterprise perspective, as much of the same basic functionality keeps getting developed over and over again, and maintenance costs are spread over a large set of redundant capabilities. Also, the continuous consumption of valuable project resources to design, develop, and test new gateways increases technical, schedule, and cost risk to user programs.

Besides the inability of federation managers to discover potentially useful gateway capabilities, there are other barriers to achieving higher levels of reuse. Even if alternative gateways are identified that are "better" (i.e., more functions, more reliable, easier to use), the costs associated with transitioning to a new gateway may be excessively high, and thus may not be practical from a business model perspective. The reason for these high costs is that each gateway tends to have unique mechanisms for defining required translations and other configuration elements, and thus the investment in time and resources needed to train developers to set up and operate the gateway properly is not something that a federation manager would want to repeat. This could result in dissatisfied users being "locked in" to special-purpose proprietary gateways, resulting in barriers to user communities migrating to more efficient gateway solutions in the future.

REQUIREMENTS FOR GATEWAY IMPROVEMENTS

Clearly, the current inefficiencies with respect to gateway employment in the DoD adversely affect the time and resources needed to develop and execute LVC events. With anticipated increases in the number of multi-architecture LVC events in the future, it is critical that enhancements to gateway user processes and products be developed. The following list describes the core requirements for addressing these challenges[5].

Requirement 1

The user shall have open access to a knowledge base of existing gateway information. The knowledge base shall have user-friendly features for users to browse

across the set of gateways offered by government or commercial providers.

Rationale
One of the main reasons that so many program managers elect to build new gateways rather than reuse existing ones is because there is no means (other than web surfing or referrals) to determine what existing gateways are available for reuse. Creating an easily accessible and well-maintained knowledge base of reusable gateways with associated Point Of Contact (POC) information would provide the visibility needed for users to make more informed "buy versus rent" decisions.

Requirement 2

The user shall have a mechanism to assess candidate gateways based on the performance characteristics of those gateways.

Rationale
In many LVC environments (particularly in the testing community), latency is a major concern. Even modest amounts of system latency may be intolerable for hardware-in-the-loop applications. When gateways are a part of the overall architecture for the LVC environment, gateway performance becomes a major concern. However, performance data is not provided in most gateway descriptions, and available data is not provided using any standard metrics or formats. A standard representation of gateway performance characteristics will allow side-by-side comparisons of gateway performance, allowing for better decisions by gateway users.

Requirement 3

The user shall have a user-friendly mechanism to describe their application-specific gateway requirements according to a standard listing of gateway capabilities.

Rationale
Once a user has insight into the range of reuse opportunities, there needs to be a way for users to describe what gateway capabilities are needed to support their immediate application. In the absence of any standard method for describing gateway requirements, users are forced to develop their own. This can easily result in gaps in the specified requirements, which may not be discovered until considerable resources have already been spent trying to get the gateway to work properly. Also, if users have no template for describing their gateway requirements, the resulting requirements may be mismatched with the way providers describe their gateway capabilities. An easy-to-use mechanism (potentially automated) that provides a standard template for users to describe gateway requirements would save time and allow direct comparisons to capabilities offered by gateway developers.

Requirement 4

The user shall have an efficient means to compare application requirements to gateway capabilities.

Rationale
Once a user has described their gateway requirements, there needs to be a way to map these requirements to the capabilities offered by gateway developers. This can be done by hand, but it may be quite tedious if the number of gateway candidates is large. Also, if there is no standard template for describing gateway capabilities in addition to gateway requirements, automated mappings of requirements to capabilities will be largely infeasible. Since the goal is to define an efficient means to support comparisons, there is an additional implied requirement for a standard template for developers to describe the capabilities that they can offer to potential users. The existence of these two templates provide the formalism needed to automate the "requirements to capabilities" matching process. Note that for machines to be able to process the data captured through these templates, the data must be described according to a machine-readable language.

Requirement 5

The user shall have an efficient, repeatable mechanism to determine what gateways best meet the application requirements.

Rationale
Once an efficient mechanism for mapping gateway requirements to gateway capabilities is established, there needs to be a way to determine, across the range of possible options, which gateway(s) provides the "best match" for the users' applications. If the matching data is available to the users, the users can make the required judgments themselves (although it can take considerable time and energy). However, implicit to the users' selections will be their own biases with respect to the relative weightings applied to each requirement-capability pairing. These biases will vary from user to user, and thus whether the optimal gateway match will really be identified will depend on who is doing the selecting. An automated approach, based on a standard set of grading rules properly socialized with the gateway developer community will not only reduce the time required to identify the optimal gateway(s), but it will also be repeatable and independent of tester bias.

Requirement 6

The user shall have an efficient mechanism to define the Simulation Data Exchange Model (SDEM) mappings necessary to meet interoperability requirements across the LVC environment.

Rationale
Once the user has selected a gateway for their application, the mappings among the different architectures and SDEM representations in the LVC event must be defined. Currently, this is all done by hand, and unique mappings must be defined for every SDEM/architecture combination. This is extremely resource-intensive, and highly subject to error. A method is needed to reduce the number of unique mappings that are required, and to define the mappings according to some standard template. Adherence to this template should reduce errors by formalizing the content, format, and syntax of mapping data, and reducing the number of unique mapping files should reduce the amount of user effort required to define the mappings.

Requirement 7

The user shall have a standard mechanism to communicate all necessary SDEM mappings and other needed translation services to the gateway in reusable files.

Rationale
Although a standard mapping template is a useful tool by itself, a greater degree of efficiency can be achieved if the process of configuring the gateway to support the defined mappings can be automated. This implies a requirement for mapping data to be machine-readable. Formalizing the format and structure of mapping files will also facilitate reuse of this data. The need for automation also implies a requirement for gateways to be able to ingest the mapping data (either directly or via an external tool) and translate the data into the internal formats used by the gateway. This capability would relieve the user of the need to configure the gateway by hand, which can be time-consuming and error-prone.

Requirement 8

The user shall have a standard mechanism to communicate all necessary gateway configuration data to the gateway.

Rationale
Most gateways are configured for a specific intended use through a configuration file. The format and content of these files vary from gateway to gateway. Gateways are usually very sensitive to the exact values defined in the configuration file, and considerable user training is generally required to be able to set up the configuration properly. Sometimes, user programs will invest in user training for their own people, and sometimes programs will just pay the developer for gateway support. In both cases, the relatively high costs of these investments create barriers to migrating to more advanced gateways that are better aligned with their needs. A standard format for gateway configuration files would result in programs having to train their people in gateway configuration procedures only once, and thus could easily migrate to other gateways without the costs of retraining.

Note that many of these requirements refer to the term "efficient." Improved efficiency in this context simply implies the need to reduce the time, cost, and technical risk associated with the methods used today. Although the characteristic "efficient" is not directly testable, the idea is to define both product and procedural improvements to current gateway methodologies, and to define an overarching process framework into which new and emerging technologies can be inserted in the future to achieve even greater improvements.

SUPPORTING PRODUCTS FOR GATEWAY SELECTION

The following provides a description of each of the emerging LVCAR-I products that collectively address the core requirements that impact gateway selection.

Gateway Capabilities Description

The Gateway Capabilities Description (GCD)[6] defines a standard set of capabilities that a gateway could potentially provide to user programs. Each capability has three elements: Capability Definition, Examples, and Levels of Implementation. The *Capability Definition* provides a concise definition of the capability. The *Examples* provide context using a real-world example. The *Levels of Implementation* indicate the degree to which the capability is supported. The number of levels varies based on the capability. Generally a level of "0" means that the capability is not implemented, and conversely, a level of "5" generally means that the capability is fully implemented. A level of "3" represents a partial implementation. Capabilities can also be defined as implementations between defined levels. A partial example drawn from the GCD is provided in Table 1.

The GCD was designed to address Requirement 3 and some aspects of Requirement 4. For Requirement 3, it

provides the formalized listing of gateway capabilities needed for gateway users to fully capture their gateway requirements. For Requirement 4, the GCD addresses the implied requirement for a standard template that developers can use to describe the gateway capabilities that they can offer to potential users. The ability to describe both gateway requirements and gateway capabilities according to the same common template establishes the foundation for direct requirements-to-capabilities mappings.

Gateway Performance Benchmarks

The Gateway Performance Benchmarks (GPB)[7] define measurable performance characteristics for gateways. The GPB is designed so that developers, testers, and consumers of gateways have a consistent set of metrics to determine which gateway or gateways will best suit the needs of the end-user. Table 2 lists the performance metric elements and means of measuring.

Table 1. Gateway Capability Description (Example)

Functional Capabilities – SDEM Translations			
Reference ID	**Capability Definition**	**Examples**	**Levels of Implementation**
FC-ST-1	Capability to perform unit conversion on a single attribute (SDEM element).	For example, if a gateway can translate meters to feet, or a similar direct algorithmic conversion.	0 = No unit conversion 1 = Single attribute conversion for 5 or fewer defined types 3 = Single attribute conversion for fewer than 15 fixed types 5 = Conversion between arbitrary units
FC-ST-2	Capability to perform complex data type conversions from single to multiple, multiple to single or different numbers of multiple attributes. This includes coordinate systems with different number of components.	For example, if a gateway can translate between coordinate systems with different number of components, such as Euler angles (3 elements) to quaternions (4 elements), or articulated parts verses single frame reference.	0 = No multiple attribute conversion 1 = Multiple attribute conversion for 5 or fewer fixed types 3 = Multiple attribute conversion for fewer than 15 fixed types 5 = Arbitrary multiple attribute coordination conversion

Table 2. Gateway Performance Metrics

Performance Metric Element	Definition	Possible Means of Measure
Resource Utilization	Loading levels for system resources:	
	• *Memory*	Percent of available megabytes or number of pages input and output
	• *Central Processing Unit (CPU)*	Percentage used for both average and maximum, and number of instructions per second required
	• *Disk*	Percentage used, and number of access operations required
	• *Input/Output (I/O)*	Number of operations for both input and output
	• *Database*	Number of database accesses per second
	• *Network*	Percentage of bandwidth used
Speed/Response Time/Latency	Time required to process inputs	Input/output response time and queue lengths (#messages/tasks
Throughput	System processing capability	Processing rate for messages, data streams, or packets
Scalability	Ability for multiple system components to process data flow efficiently	Multiple system tested using parameterized filtering
Endurance/Robustness/Stability	System component reliability and	Mean time between failures
Performance-Related Accuracy	Minimizing output errors that are due to performance characteristics	Percentage of correct output data

The GPB also defines a structured set of use cases that collectively define a range of "typical" application types (e.g., large virtual training event, small faster-than-real-time constructive event, or hardware-in-the-loop event) to which the benchmarks can be applied. These use cases are based on a defined set of scenario and operational parameters. Scenario parameters describe characteristics of the simulation, while operational parameters describe the operating environment for the simulation hardware and software.

The GPB was designed to address Requirement 2, with some indirect benefits that partially address Requirement 4. Using the GPB, users will now have a way to assess gateways not only in terms of what capabilities they provide, but also in terms of how well

they perform in defined contexts. The GPB will also be incorporated into the common language for describing gateway capabilities (see next product). Note that, in the longer term, gateway developers will have the details of the performance metrics that are sought-after by end-users, which will in turn drive the priority of improvements of future gateways and their upgrades.

Gateway Description Language

The Gateway Description Language (GDL)[8] provides a mechanism to document user gateway requirements and capabilities of gateways provided by the developer. The GDL is based on the GCD and the GPB. The purpose of GDL is to communicate user needs and developer capabilities to support the selection of gateways. GDL is implemented in Extensible Markup Language (XML) to be both human and machine-readable. GDL can be used in two ways; to specify user requirements for gateway capabilities, and to specify the capabilities offered by gateway products.

GDL has three major components: Description, Capabilities, and Performance. The *Description* identifies the creator of the GDL file and the purpose for which it was developed. Each GDL file has exactly one Description. A gateway user would create a GDL file to document the requirements for a specific distributed simulation application. This includes the name of the user (generally an organization) and the name of the simulation environment. A gateway developer-generated GDL file includes the name of the gateway and version information.

The next component of GDL is the *Capability*. A GDL file has one or more Capabilities. A gateway user-generated GDL file would capture the required capabilities for the gateway in a particular event. A gateway developer-generated GDL file would identify the implemented capabilities for a particular gateway. Each Capability entry has four components: Capability Identifier, Capability Description, Implementation Level, and Priority Level.

The final component of GDL is *Performance*. A GDL file has zero or more entries for Performance. Each Performance entry is associated with a use case. This component includes a use case identifier and the values of the performance metrics. The performance metrics for a user-created GDL represent the user's desired performance for each parameter. In a developer-created GDL, these values result from executing the GPBs.

GDL was designed to address Requirements 4 and 5. GDL provides a common machine-readable format for describing both user gateway requirements and gateway capabilities, as defined by developer organizations. Although matching requirements to capabilities can be done by hand (since GDL is also human-readable), the common format greatly facilitates the automation of this process. This automation extends to the determination of "best matches," as standard rule sets can be developed, socialized with gateway developers, and applied to help identify those gateways that most closely align with defined user requirements. Thus, GDL is critical to automating the process of gateway selection, and automation is the key to achieving the desired efficiency gains.

SUPPORTING PRODUCTS FOR GATEWAY CONFIGURATION

The following provides a description of each of the emerging LVCAR-I products that collectively address the core requirements that impact gateway configuration.

SDEM Mapping Language

The SDEM Mapping Language (SML)[9] provides a formal XML schema for defining required translations between SDEMs. This schema is independent of any specific gateway implementation. The SML format allows the elements of one SDEM to be mapped to elements in another SDEM, including any additional required transformations. While some gateways have a method for describing translations between SDEMs, none are complete or formally defined as a community standard. SML fills that gap by providing a means for SDEM mapping files to be reusable from application to application.

Choosing XML as the basis for SML addresses three major objectives of the mapping language: human-readable, machine-readable, and existing tool support. One of the key benefits of using SML to create an SDEM mapping file is that it provides formal documentation at the detail level for the translations between SDEMs. This formal definition of the required translations allows for all of the participants to review and agree to the translations. Once they are agreed to, the mapping file becomes a "contract" between the event leadership and the gateway provider. Because SML is also machine-readable, the gateway provider may choose to have their gateway directly use SML as a means to configure the translations. Using SML allows users to take advantage of existing tools for creating, viewing, importing, and parsing XML files.

SML is composed of a series of *ElementMaps*. The ElementMap has three components: FromElementName, ToElementName, and Transformation. The *FromElementName* component is the name of the data construct to be translated. The *ToElementName* component is the name of the data construct to be translated to. The *Transformation* component defines the steps to convert the data from one SDEM to the other.

SML was primarily intended to address Requirement 7. Developers of multi-architecture distributed simulation environments perform the mapping process by creating a defined set of mapping files which define the persistent and transient objects to be shared among the event participants. Capturing this data in SML allows the event participants to review and verify all required translations, independent of the gateways selected. The formality of the SML specification also makes it possible to verify the completeness and consistency of the translations. Although SML is useful as a standalone language, most users are expected to take advantage of the fact that SML is XML-based, and apply the many tools that are available to view and manipulate XML files. SML also streamlines the gateway configuration process by providing a machine-readable format that can be ingested by gateways. This can save considerable time and effort compared to the manual process of converting raw mapping data to the internal formats used by the selected gateway.

ANDEM

Although gateway mapping procedures benefit greatly from the use of SML, the process of producing mappings across all the different architectures and SDEMs being used in a distributed simulation event is still a very labor-intensive, manual process. This process is illustrated in Figure 1, which depicts a mapping between architectures and data exchange models in a notional event/exercise. Each exercise participant in this example has to create mappings to and from every other architecture/SDEM in the exercise, in this case resulting in twelve unique mappings.

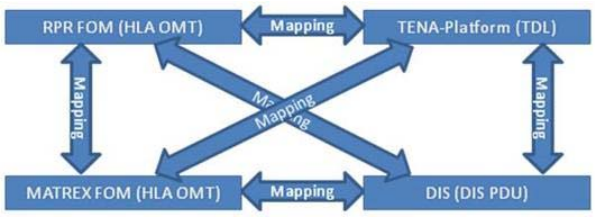

Figure 1. Current Mapping Procedures

Figure 2 illustrates how a common Architecture Neutral Format (ANF) can reduce the time and effort required for SDEM mapping. In this situation, the number of unique mappings is decreased from twelve to eight. An additional benefit is that the likelihood of inconsistencies across the mappings is significantly reduced due to the smaller set of mappings to consider.

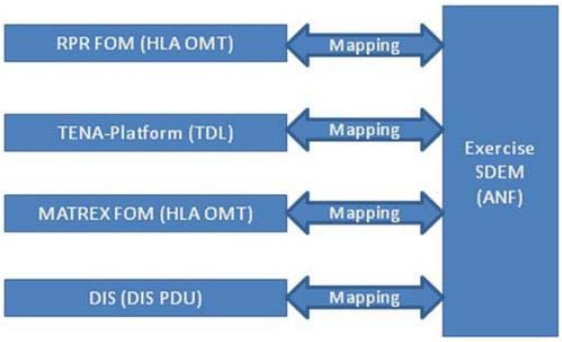

Figure 2. ANF-Based Mapping Procedures

A product called the Architecture Neutral Data Exchange Model (ANDEM), developed under Joint Training Integration and Evaluation Center (JTIEC) sponsorship on a separate project, defines the desired ANF[10]. The use of ANDEM directly addresses Requirement 6, as the event/exercise-wide SDEM represented in ANDEM would define the contract for data exchange to which all participants would map their local SDEM representations.

In addition to using ANDEM as a way to reduce the time and effort required to formulate the necessary mappings, ANDEM is also useful during gateway configuration. As mapping files are produced, even greater efficiencies can be achieved if the mappings can be input to the gateway in an automated fashion. However, if the mapping files are based on translation of architecture-specific SDEMs to ANDEM, this requires the selected gateway to properly translate the ANDEM representation to its internal database format. Although this translation can be performed by front-end tools external to the gateway, it may also be possible for gateways to adopt the ANDEM format for internal storage, which would allow them to ingest SDEM mapping files directly in ANDEM format. In either case, use of ANDEM should facilitate the process of capturing all of the necessary mapping information in the chosen gateway.

Gateway Configuration Language

The Gateway Configuration Language (GCL)[9] is designed to capture common gateway configuration data in a single implementation-independent format

that is both human and machine-readable. Therefore, like the other gateway languages, GCL is based on XML. This implementation allows the user to directly review the file, and allows gateways and other tools to directly read it. GCL also provides documentation on the use of a gateway for a specific purpose. Some of the fields may not be needed by the gateway implementations, but can be included for documentation reasons.

GCL is composed of three main components: General Data, Architecture/SDEM Interface Description, and Filters. The *General Data* section contains data related to the overall gateway execution and information about the use of the gateway, such as the name of the gateway and the event to which it is to be applied. The *Architecture/SDEM Interface Description* defines a separate record for each architecture/SDEM combination or "side." All gateways have at least two sides, and some gateways may have more. The side record defines the architecture-specific parameters used by the gateway to connect to the architecture. For instance, Federation Execution Details (FED) file names and Runtime Infrastructure (RTI) Initialization Data file names would be included for HLA elements, while *emEndpoints* and *listenEndpoints* would be included for TENA elements. The side record also contains information about the selected SDEM (name and version) and the name of the SML file used to map the side's architecture/SDEM to the common representation used by the gateway. The *Filters* section is used in circumstances where gateways are used to provide filtering that is not supported by the architecture or to create enclaves. GCL provides an implementation-independent format for defining filters.

GCL was intended to address Requirement 8. The human-readable aspect of GCL makes it easier for LVC developers to review and comment on the configuration of the gateways prior to the event, as compared to gateway implementation-specific configuration files. The use of a common format for gateway configuration files reduces the time and cost associated with gateway configuration activities, and thus facilitates having developers choose the gateways that best meet application needs rather being restricted to those gateways (with non-standard configuration file formats) that they already know how to use. In addition, GCL files are fully reusable, and thus can reduce the time and costs associated with future LVC events that employ the same gateways.

IMPROVED PROCESSES AND TOOLS FOR GATEWAY SELECTION/CONFIGURATION

To take full advantage of the new gateway products described in this paper, improvements to the older processes used to select and configure gateways are also necessary. The enhanced gateway selection process illustrated in Figure 3 begins with the gateway developers, who describe the capabilities that their product(s) provide in GDL notation. These GDL descriptions from various vendors are then put under the stewardship of an appropriate management organization (addressing Requirement 1). When requirements for new LVC events are defined, users access this knowledge base of gateway capabilities and search for the gateways that best match their requirements. If several gateways are found that meet the requirements of the LVC event, some amount of electronic or face-to-face discussion between users and developers may be necessary to down-select to the gateway or gateways that best meet the customer's needs.

The enhanced gateway configuration process illustrated in Figure 3 begins with the development of the required mappings using SML. Note that fewer mappings will be required due to the use of ANDEM as the architecture-neutral format that all SDEM representations will map to, and that reuse opportunities for SML files used in prior LVC events may be available. Once the mappings are defined, the GCL standard content/format is used to produce the gateway configuration file needed by the selected gateway(s). Again, reuse opportunities associated with existing GCL files should be taken advantage to the greatest degree possible. Finally, the content of the GCL file is input to the gateway and tested during the integration of the full LVC environment, to detect any anomalies before execution.

Although this process does not depend on the availability of supporting tools, automation will be critical to achieving defined efficiency goals for gateway employment in future LVC events. The tools that support the efficient application of these new gateway products are shown in Figure 3 at the points in the process at which they are most relevant. The first tool used in this process is the *GDL Editor*. This is a tool specifically designed to create GDL files. The tool interface is interview-based, much like modern tax software packages. The questions that are posed are based on the content of the GCD. Developers use this tool to create GDL files that describe the capabilities that their gateway can provide. Users employ this tool to define the gateway capability requirements for their

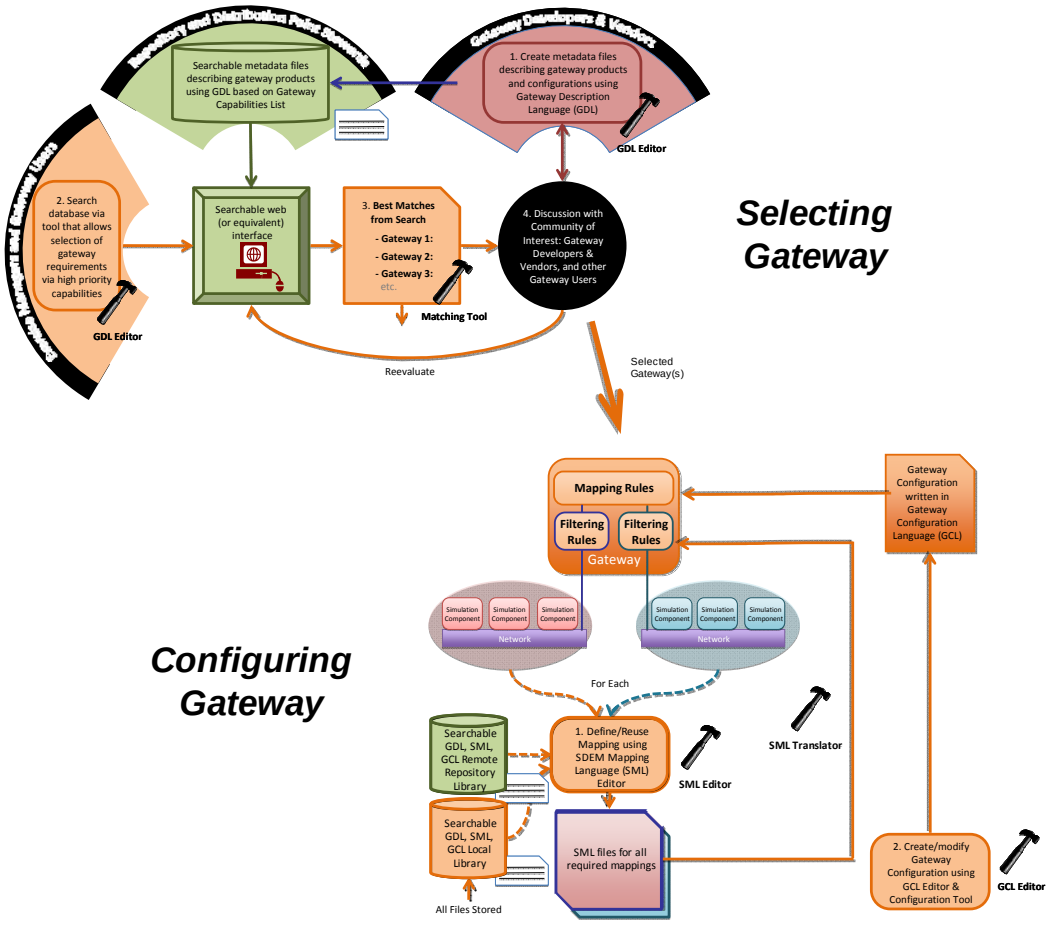

Figure 3. Improved Processes and Tools for Gateway Selection and Configuration

application. The output of the tool is a file compliant with the GDL schema.

Once developer-provided GDL files are made available, a mechanism is required for users to quickly and easily view what gateway capabilities are available for reuse. This knowledge base is implemented as a repository, with the requisite browsing and searching capabilities. Users can exploit this centralized "virtual warehouse" to discover the full breadth of options for supporting gateways. These browsing/searching capabilities are "tuned" to support GDL, since users will typically want to search for gateway capabilities that are embedded in the archived GDL files.

Although users will be free to browse for gateways directly via the repository, a *Matching Tool* is provided that can read a user-defined gateway requirements file in GDL, compare these requirements against the capabilities offered by gateway developers (also expressed in GDL), and return "best matches." The determination of best matches depends on an internal algorithm that determines the closest match of defined requirements to supporting gateways. The tool produces a set of best gateway matches for user consideration. The user can either accept the tool's recommendation, choose an alternate gateway (perhaps based on additional relevant factors per discussions with gateway owners), or modify their requirements and restart the matching process. This tool was explicitly designed to address Requirement 5.

The next two tools are the *SML Editor* and *GCL Editor*. The SML Editor is a tool specifically designed to create SML files. Users are presented with a front-end template for capturing the required mappings, which the tool converts into the SML format. The GCL Editor follows the same concept, with a user-friendly interface to translate gateway configuration data into GCL format for input into the selected gateway(s). Note that for both of these tools, local and remote repository capabilities may assist users in identifying reuse opportunities for existing SML and GCL files, which can then be tailored to the application at hand via the defined editing tools.

The final tool designed to support the gateway selection/configuration process is the *SML Translator*.

As discussed previously, significant efficiencies can be gained by basing all required mappings on the ANDEM architecture-neutral format rather than specifying all mappings on a point-to-point basis. It is also very desirable to have the resulting SML files ingested into the gateway without significant human intervention, as SML is fully machine-readable. However, if the mappings defined in the SML files are based on ANDEM, the gateway is unlikely to know how to convert the mappings into the architectures used by the simulations within that particular LVC event. Thus, a means to convert from the ANDEM-based mappings (described in SML) into the architecture-specific mapping data needed by the gateways is required. One possible means of doing this conversion would be to have the supporting gateway(s) use ANDEM internally as the local database format. However, most gateway developers are not interested in basing their internal formats on an external product that they do not control, and which may be less efficient than existing internal formats. Thus, this tool was designed to translate from the ANDEM-based mappings in SML to the internal mapping formats needed by the supporting gateway. This tool produces mapping file(s) that can be directly ingested into the gateway. The gateway uses this file to self-configure its required mappings in preparation for the LVC event execution.

NEXT STEPS

The JTIEC-sponsored LVCAR-I effort is scheduled to be completed in the fall of 2012. This phase of the effort has produced numerous products, such as language specifications, tools, and supporting processes. During FY12, much of the focus will be on community outreach, in order to identify additional user requirements for gateway products and to obtain user/developer feedback on existing products (leading to product updates). Concepts to be discussed include the possible establishment of an independent Gateway Testing Laboratory for verifying claims made by developers about their gateway products, and the potential of taking some/all of the gateway language specifications into formal standardization. It is anticipated that these discussions will identify the need for follow-on activities, potentially supported by different sponsors.

SUMMARY

This paper has described efforts to introduce a new level of systems engineering rigor to the process of selecting and configuring gateways in support of the development and employment of multi-architecture distributed simulation environments. A wide range of supporting products have been developed to enable the activities within this evolved view of process, including new gateway languages, tools, and benchmarks. These new products are immediately available to DoD users, and may become more generally available to other gateway users in the near future. It is believed that adoption of these products will streamline gateway selection/configuration activities, reduce the technical risk associated with the employment of gateways in the simulation environment, and generally allow multi-architecture distributed simulation environments to be constructed "better, faster, cheaper" in the future.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Gary Allen of JTIEC for his sponsorship and overall leadership of the LVCAR-I effort, and to Dr. James Coolahan of JHU/APL for providing the program management oversight needed to successfully execute the LVCAR-I Gateways activity.

REFERENCES

1. Lutz R., Drake D., "Gateway Concepts for Enhanced LVC Interoperability", 2011 Spring Simulation Interoperability Workshop (SIW), 11S-SIW-024, April 2011.
2. Live, Virtual, Constructive Architecture Roadmap (LVCAR) Final Report, Institute for Defense Analyses, September 2008.
3. Live-Virtual-Constructive Architecture Roadmap Implementation, Common Gateways and Bridges Characterization Report, JHU/APL NSAD-R-2010-031, May 2010.
4. Live-Virtual-Constructive Architecture Roadmap Implementation, Common Gateways and Bridges Execution Plan, JHU/APL NSAD-R-2010-049, June 2010.
5. Live-Virtual-Constructive Architecture Roadmap Implementation, Common Gateways and Bridges Task – Gateways Configuration Model, JHU/APL NSAD-L-2011-083, April 2011.
6. Live-Virtual-Constructive Architecture Roadmap Implementation, Common Gateways and Bridges Task – Gateways Capabilities Description, JHU/APL NSAD-R-2010-100, November 2010.
7. Live-Virtual-Constructive Architecture Roadmap Implementation, Common Gateways and Bridges Task – Performance Benchmarks, JHU/APL NSAD-R-2011-0XX, January 2011.

8. Lessmann K., Cutts D., O'Connor M., "LVCAR Enhancements for Selecting Gateways", 2011 Spring Simulation Interoperability Workshop (SIW), 11S-SIW-054, April 2011.
9. O'Connor M., Cutts D., Lessmann K., "LVCAR Enhancements for Using Gateways", 2011 Spring Simulation Interoperability Workshop (SIW), 11S-SIW-025, April 2011.
10. "ANDEM RDF/XML Schema," Live-Virtual-Constructive Architecture Framework (LVCAF) Report, 2010.