LVCAR Enhancements for Using Gateways

Michael J. O'Connor
ITT Corporation
6767 Old Madison Pike
Suite 160
Huntsville, AL 35806
256.964.1470
michael.oconnor@itt.com

Dannie E. Cutts
The AEgis Technologies Group, Inc.
410 Jan Davis Drive
Huntsville, AL
256.799.1154
dcutts@aegistg.com

Kurt Lessmann
Trideum Corporation
4946 Research Drive
Huntsville, AL
256.704.6116
klessmann@trideum.com

Keywords:
LVCAR, Gateway, GDL, GML, GCL

ABSTRACT: *The Live Virtual Constructive (LVC) Architecture Roadmap (LVCAR) report, sponsored by the Modeling and Simulation (M&S) Steering Committee, recommended several actions to promote the reuse of common M&S capabilities and to foster common formats and policy goals to promote LVC interoperability. One of these recommended actions pertained to the development of common gateway capabilities. The first phase of this effort reviewed the gateway needs of users and developers to develop strategies to support more effective LVC integration using gateways. The current LVCAR phase is focused on the implementation of the selected "Enhance" strategy, which emphasizes improvements to the native methodologies and practices LVC engineers follow to select and configure gateways for their specific applications. This paper describes the products intended to support using gateways.*

The two products are the Gateway Mapping Language (GML) and Gateway Configuration Language (GCL). These products are intended to support the end user in getting more value out of the gateways they use as well as making it easier to switch gateway implementations. GML is a formal language for mapping the required translations between Simulation Data Exchange Models (SDEMs). This mapping is required for all multi-architecture events but there is currently no way to formally specify the required translations. In most events creating the required translations is done by trial and error with no resulting documentation. GML provides the format for this documentation and allows users and providers to agree on the required translations. GML also produces formal documentation that makes it easier to use the same translations with other gateways in the future. GCL is a formal language for specifying common gateway configuration parameters. In addition to common parameters, GCL includes support for many exercise management features such as data filtering. GCL allows a federation to document these key parameters in a gateway implementation independent format. This defined format supports reuse of the information in future events.

1. Introduction

Distributed simulation is a well established approach for addressing many simulation problems. The rise of distributed simulation has resulted in the development of a number of distributed simulation architectures. Different communities and users have chosen their architectures based on their individual needs. Some of the architectures also support the creation and use of different Simulation Data Exchange Models (SDEMs), which specify how data is to be shared between simulation components. There are many occasions when it becomes necessary to link simulations that use different architectures and/or SDEMs. This situation resulted in the creation of gateways/bridges (this paper will use the single term gateways) between the different architectures/SDEMs. However, the use of gateways presents a formidable challenge to the simulation community.

The use of gateways was one issue that led to the creation of 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 [1] and supporting documentation that collectively totaled over one thousand pages. The implementation of LVCAR recommendations began in the spring of 2009, and is currently in progress. One of the key actions from the LVCAR Final Report focused on improvements to gateway and bridge capabilities. The LVCAR Gateways and Bridges implementation task was designed to address the issues of gateways and, more generally, to provide the LVC user community with improved mechanisms for gateway discovery, configuration, and employment. [2]

2. Terminology

One of the challenges of the gateways project is the lack of standard names and definitions for the associated distributed simulation concepts. Distributed simulation was originally created to meet an urgent need. While a lot of thought and engineering went in to the development of distributed simulation applications and protocols, less effort was given to the definition of the overarching concepts. This lack of definition makes the discussion of multiple architectures difficult. While almost all distributed simulation architectures implement similar concepts, there tends not to be concise naming schemes for those concepts. An example of this situation is the Distributed Interactive Simulation (DIS) protocol. Most anyone with distributed simulation experience will recognize the layout of a DIS Protocol Data Unit (PDU) in table form. However, the DIS PDU table format has no name or formal definition. Recently the SISO DIS Product Support Group (PSG) has made some effort in regularizing the table format, but they have still not created a standard for the format.

There are multiple mechanisms for sharing information using distributed simulation. These mechanisms are referred to by several terms including protocols and architectures. Examples include DIS, the High Level Architecture (HLA), Test and Training Enabling Architecture (TENA), and Common Training Instrumentation Architecture (CTIA). For this discussion we will refer to these as architectures. When we use the term architecture we are referring to the mechanism used to share the data. Some of the architectures, DIS and CTIA, have a single data model. When we refer to DIS as an architecture, we mean the data transport not the data content.

Because gateways are inherently multi-architecture in nature, this lack of a naming convention for reoccurring concepts makes gateway discussions difficult. Therefore before a discussion of our new gateway products can begin, some existing concepts must be defined. Since the introduction and use of new terminology and acronyms can at times be confusing, this section will define these concepts before they are used in the description of the new gateway products.

All distributed simulation architectures define the data to be shared among the participating simulations. In DIS, the data was specified as Protocol Data Units (PDUs). The High Level Architecture defined the term "object model" to include both Simulation and Federation Object Models (SOMs and FOMs) respectively. This specification occurred despite the fact that the term “object model” is also associated with the software engineering concept, object-oriented. While DIS PDUs and an HLA Object Models are clearly analogous concepts, there is no general terminology available to describe the idea. (TENA and CTIA implement similar concepts). To represent this data sharing concept we use the term Simulation Data Exchange Model (SDEM). A SDEM is the representation of information about persistent and transient objects shared via distributed simulation. A SDEM has three components:

- Format (specification) – describes how to represent the information;
- Data Construct – description of the persistent and transient objects and their attributes in the specified format;
- Semantics – Additional information relating to the publishing and receiving of objects.

An example of a SDEM is the Real-Time Platform Federation Object Model (RPR FOM). The Format is formally defined in the HLA Object Model Template (OMT). The Data Construct is the set of persistent and transient objects defined in the FOM which is in the OMT format. The Semantics for the RPR FOM are defined in the RPR Guidance and Interoperability Rationality Manual (GRIM).

We have identified three types of SDEMs: Neutral DEM (NDEM), Mapping DEM (MDEM), and Architecture Specific DEM (ASDEM). These types are relevant to the operation of the gateway.

- Neutral SDEM (NDEM) – A SDEM whose format is not associated with any distributed simulation architecture.
- Mapping SDEM (MDEM) – A SDEM used to map elements from one SDEM to another SDEM.
- Architecture Specific SDEM (ADEM) – A SDEM whose format is defined as part of distributed simulation architecture.

Each type of SDEM has a unique Format.

- Architecture Neutral Format (ANF) - A SDEM format for representing data exchange information that is not tied to specific distributed simulation architecture
- Architecture Specific Format (ASF) – A SDEM format for representing data exchange information that is tied to specific distributed simulation architecture

- Architecture Mapping Format (AMF) – A SDEM format that allows the elements of one SDEM to be mapped to elements another SDEM (the mapping may be ANF-to-ANF, ASF-to-ASF, or ANF-to-ASF).

A distributed simulation event requires a SDEM and an architecture. This requirement will be referred to as an architecture/SDEM pair when the general concept is discussed. An example of an actual architecture/SDEM pair would be HLA/RPR FOM.

Most distributed simulation users are only familiar with ASDEMs and ASFs. Common examples of ASDEMs include the collective set of DIS PDUs, HLA RPR FOM, HLA MATREX FOM, and TENA TENA-Platform. As discussed above, the ASF for DIS is not documented. The developers of HLA and TENA created formal ASFs for their architectures, OMT and TENA Definition Language (TDL) respectively. To date there are no formally developed NDEMs, but there is an ANF – Architecture Neutral Data Exchange Model (ANDEM) [5]. ASFs and ANFs are similar but do differ. ASFs will include representations for specific features and constructs of the architecture they are associated with, such as Data Distribution Management (DDM) in HLA and Remote Method Invocation (RMI) in TENA. An ANF may support the representation of similar concepts, but will likely be more abstract than the ASFs.

While the concepts of ASDEMs and ASFs are well understood, and the concepts of NDEMs and ANFs can be understood based on the former, MDEMs and AMFs are more difficult to understand. Only ASDEMs and ASFs are needed for events using single architecture/SDEM pair. NDEMs and MDEMs are only applicable for simulation events using multiple architecture/SDEM pairs. An NDEM can be said to be the interoperability intersection of the multiple ASDEMs that are used for an event. Although most events do not currently define a formal NDEM, there is at least some understanding of the interoperability intersection. Much less defined and understood is the formal mapping of SDEM elements. This is where the MDEM and AMF come into play. The MDEM maps elements from one SDEM to another, including the required translation. The AMF allows the association of the named elements with the translation. So while ASDEMs and NDEMs may look similar, the MDEM looks very different.

3. Problem Statement

As part of the LVCAR Gateways and Bridges implementation task, two broad areas of user need were identified [3]. The first was to help the user select the best gateway for their needs. The second was to help the user to enhance the use of gateways. This paper addresses the enhancements to using a gateway. The tools developed to support the selection of gateways are the topic of another paper. [4]

After navigating the selection process, the user is left with two main concerns: lack of documentation on the configuration of the gateway, and being locked into a single gateway implementation. The second issue is a result of the first. The documentation issues fall into two primary categories: architecture/SDEM mapping and gateway configuration. There is currently no standard method for mapping the elements of one architecture/SDEM pair to another. While in most events some attempt is made to map the architecture/SDEM pairs, the lack of a formal method for the process makes it difficult. Most current processes do not produce a sufficient artifact for future use. The lack of a standard set of configuration parameters also makes documentation difficult. The exact mappings and configurations are generally arrived at by trial and error in a spiral testing process but are not documented.

The lack of documentation for the requirements and objectives of an event or exercise leads to the perceived lock-in for a particular gateway implementation. Because the working set of mappings and configurations were arrived at through trial-and-error, the user may have a working solution but does not know what it is. Therefore the user would have to repeat the time consuming process to configure a new gateway.

The following sections demonstrate how the proposed Gateway Mapping Language (GML) and Gateway Configuration Language (GCL) address these user issues.

One of the challenges presented to the LVCAR Gateways Team is defining the benefits to the end user of the task. To better facilitate the discussion, the benefits to an example user will be described. The challenges faced by our example user will be presented, followed by an overview of how the products developed by this effort will provide value. This discussion will begin by defining the problem the user is trying to address with gateways. The example in this case is a medium complexity multi-architecture LVC event. While all gateway users from the simplest to the most complex will receive some benefit from the products described herein, a more complex example was chosen to more fully illustrate the range of benefits.

Our example user is the event manager for a LVC event involving multiple organizations. The exact nature of the event is not important for our example. The event manager, working with the participating organizations, selected the simulations required to meet the objectives of the event. In some cases existing federations were

selected and in other cases individual simulations were specified. The selected simulations and federations use different architectures and SDEMs. For various reasons including cost, schedule, and organizational considerations, it is not feasible to convert all of the selected simulations to use a single SDEM and architecture.

Our example event is spread over six sites and uses four different SDEM/architecture pairs: DIS, HLA/RPR FOM, HLA/MATREX FOM, and TENA/TENA-Platform. Three sites will only use two of the SDEM/Architecture pairs, DIS and HLA/RPR FOM. Two sites will use three SDEM/Architecture pairs, DIS, HLA/MATREX, and TENA/TENA-Platform. Figure 3.0-1 shows the site configuration. The primary control site will use all four SDEM/Architecture pairs. In addition to SDEM translation issues, there are other exercise management requirements. For this example, we will assume the event has created an architecture neutral SDEM in an architecture neutral format. The process for creating a NDEM is beyond the scope of this discussion.

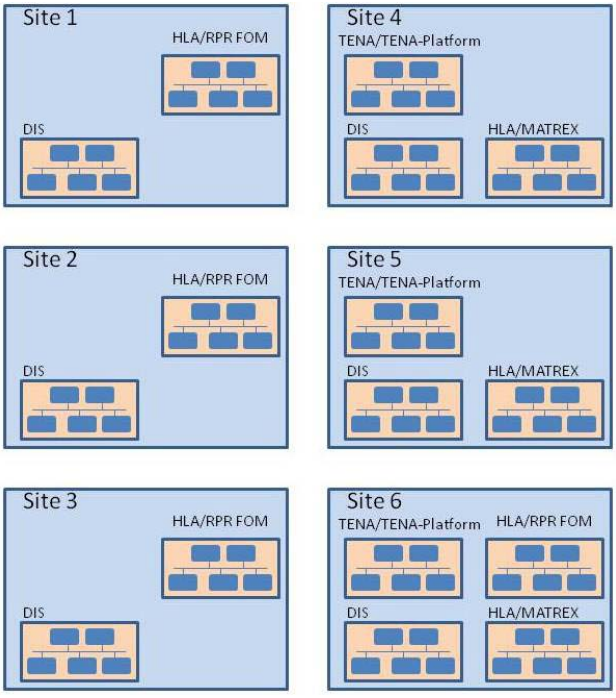

Figure 3.0-1 User Example

The event manager has two major gateway issues: 1) how to select a gateway; and 2) how to use the gateway. The process for the selection of a gateway is covered elsewhere [4].

In most large events, more than one gateway is typically used making this gateway configuration challenge more difficult. The event manager now has to set up the required SDEM translations. Creating a NDEM for the event has simplified the problem, but not solved it. The translations will most likely be implemented by the gateway provider. It is possible a text based document will be created to define the required translations. However, lacking a formal language for defining the translations will make this difficult. In most events, the required transformations are never documented in a gateway independent format, making review by the event team difficult. In most cases, the translations will only be refined by executing the event and fixing what does not work. In the end, no documentation will be created and the event manager will be tied to a specific gateway. The event manager will encounter a similar problem when configuring the gateway. No gateway independent gateway configuration language exists. This will particularly be an issue for any exercise management requirements.

4. Gateway Mapping Language

The Gateway Mapping Language (GML) provides a mechanism to define the required translations between SDEMs that is independent from a specific gateway implementation. Some gateways have a method for describing translations between SDEMs, but none are complete or formally defined as a community standard. GML is useful as a standalone language, but the development of tools based on GML will provide additional benefits.

4.1 GML Definition

There are three types of formats for SDEMs: Architecture Specific Format (ASF), Architecture Neutral Format (ANF), and Architecture Mapping Format (AMF). GML is an AMF. An AMF is defined as “A format that allows the elements of one SDEM to be mapped to elements another SDEM including any required transformation”.

GML is a format used to create a Mapping Data Exchange Model (MDEM). Like all SDEMs an MDEM is composed of a format, data construct, and rules. A MDEM is used to define the translation between one SDEM and another. A MDEM can be used to map between two ASDEMs or between an ASDEM and a Neutral Data Exchange Model (NDEM). GML supports either case. A MDEM is required for each pair of SDEMs.

GML is implemented as a XML schema. Choosing XML as the basis for GML meets three major requirements: human readable, machine readable, and existing tool support. One of the key benefits of using GML to create a MDEM 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 MDEMs become a "contract" between the event leadership and the gateway provider. Because GML is also machine readable, the gateway provider many choose to have their gateway directly use GML as a means to configure the translations. Using XML to create GML allows all of the users to take advantage of existing tools. There are many tools for creating, viewing, and importing XML files.

Figure 3.1-1 provides a notional example of what the GML schema may look like. GML is composed of a series of ElementMaps. The ElementMap has three components: FromElementName, ToElementName, and Transformation. The FromElementName is the name of the data construct to be translated. The ToElementName is the name of the data construct to be translated to. Both of these names are in the form defined by their respective ASF or ANF. The Transformation component defines the steps to convert the data from one SDEM to the other.

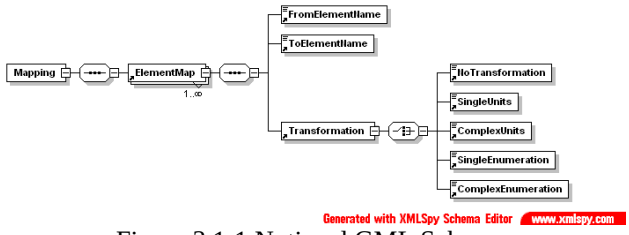

Figure 3.1-1 Notional GML Schema

4.2 GML User Applicability

Based on the Gateway Description Language (GDL) and Gateway Performance Benchmark results, our user selected one gateway for the DIS to HLA/RPR translations and another gateway to perform the multiple SDEM/Architecture translations. The DIS to HLA/RPR gateway supported specific exercise management functions and performance required for those sites.

Without GML, our user would not have a formal way to specify the required translations between ADEMs. Using traditional methods, the event team would have to create twelve mappings (mappings are one way) as shown in Figure 3.2-1. The event team may have produced a series of text documents to describe the translations required, however there would be no easy way to verify completeness. Also, there would not be an easy way to ensure consistency between the mappings. For example, the translation of the same concept from DIS to HLA/RPR may be correct, but the translation from DIS to HLA/MATREX may be wrong. This type of error would be very difficult to identify. The gateway developers or users (depends on the gateways extension model) would be left to implement the required translations based on incomplete information. In this case, the user needs two gateways to do some of the same translations. There is no mechanism to insure that both gateways perform the translations the same way.

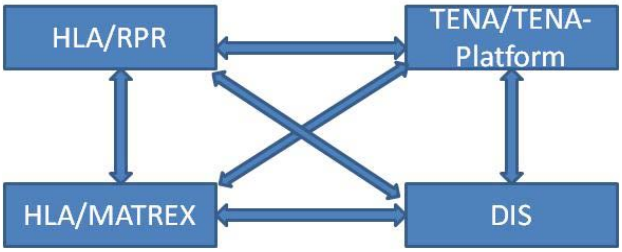

Figure 3.2-1 Notional point to point mapping

The innovation of GML along with the concept of NDEMs provides the needed structure to address the user's problem. To begin, the event team creates a NDEM that represents the interoperability intersection (the persistent and transient objects to be shared) among the four Architecture/ADEM pairs that is required to meet the event goals. The team uses the ANF ANDEM to create their NDEM. The creation of the NDEM is beyond the scope of this paper. As shown in Figure 3.2-2, the user would only have to create eight mappings between the selected ADEMs and the NDEM used for the event. The MDEMs developed using GML will allow the full event team to review the required translations independent of the gateways selected. This is particularly valuable when the event will use more than one gateway. The formal specification of the event NDEM and the MDEMs for each ADEM make it possible to verify the completeness of the translations. The chance of inconsistencies is reduced because a single translation is defined for each ADEM. The same translation specification as defined by the MDEM can be provided to both gateway developers. This reduces the chance that translations will be performed differently. Because the MDEMs are specified in GML, which is XML based, most users will already have tools to view the MDEMs.

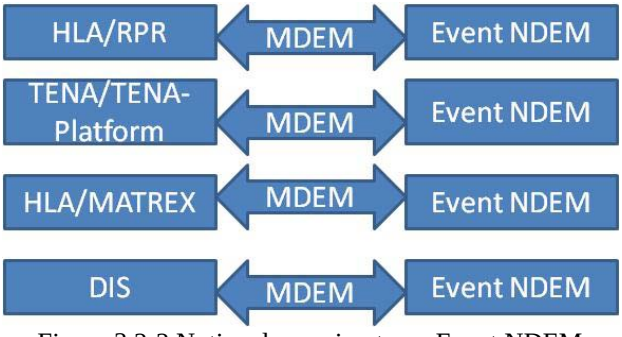

Figure 3.2-2 Notional mapping to an Event NDEM

GML has provided our user a mechanism to define and document the required translations between multiple ADEMs. This has provided a way to distribute and

review the translations independent of the gateway implementation. The approach of using GML, along with a NDEM, has reduced the number mappings required for our example event. The formal specification has also reduced the chance for errors resulting from misunderstandings from verbal or plain text discussions. Overall, GML has reduced the work required for a complex event as well as reduced the risk.

4.3 Future Tools Based on GML

The above benefits will be available to users immediately. There are additional future benefits. In the above scenario, the user had to create the MDEM using just general purpose XML editing tools. In the future a specialized MDEM creation tool would be available. This tool would simplify the creation of the MDEM and provide additional automated tools for verify the translations. Tools could also be developed to test gateways using GML as the test description. As GML is adopted by event manages, gateway developers would modify their tools to directly use MDEMs created in GML to configure their gateways.

4. Gateway Configuration Language

The Gateway Configuration Language (GCL) provides an implementation independent mechanism for documenting the configuration of a gateway. GCL allows the user to define key common gateway parameters independent of a specific gateway implementation. GCL also includes a format for defining exercise management configuration. An independent format like GCL enables users to more easily switch from one gateway implementation to another.

5.1 GCL Definition

The purpose of GCL is to capture common gateway configuration data in a single implementation independent format. As with other languages developed for the gateways project, the language needs to be both human and machine readable. Therefore, like the other languages, GCL is based on XML. This implementation allows the user to directly review the file, as well as gateways and other tools to directly read it. An important aspect of GCL is to provide documentation on the use of a gateway for a specific purpose. Some of the fields may not be needed by the gateway implementations, but are included for documentation reasons.

GCL is composed of three main components, general data, architecture/SDEM interface description, and filters. The general data element contains data that applies to the operation of the gateway and is not dependent on the architecture/SDEM. There will be an architecture/SDEM interface description record for each architecture/SDEM being translated by the gateway. The filter element is used to define exercise management type filters. It is expected that individual gateway implementations will require some specific data in addition to what is contained in GCL.

There is exactly one general data section per GCL file. The general data section contains data related to the overall gateway execution and information about the use of the gateway. This section contains more information for documentation purposes than the other two sections. The gateway purpose and event name elements are intended for documentation. These fields allow the user of the gateway requirement to document their specific needs. Some events use a single gateway, others use multiple. The gateway purpose allows the user to state the use of the gateway using this GCL to distinguish it from other gateways used by the event. The owner name is person or organization that owns the requirement for this gateway.

The general data section does contain some data that may be used directly by the gateway implementation. Examples of this include debug level and Graphical User Interface (GUI) options. Additional general gateway parameters will be included in this section.

The next section defines each architecture/SDEM interface of the gateway. For economy of language these are referred to as “sides”. A side is one architecture/SDEM pair that is translated by the gateway. All gateways have at least two sides and some gateways may have more than two sides. The sides may have the different architectures and SDEMs or even the same architecture/SDEM pair on each side. Each side is named. This is done for documentation purposes and to support the filter section described later.

The side record defines the architecture specific parameters used by the gateway to connect to the architecture. For HLA this would include elements for the FED file and RID file names. For TENA it would include specification of the emEndpoints and listenEndpoints. For DIS the port number and exercise id. This section will also contain other parameters needed for translation based on the architecture.

The side record also contains information about the selected SDEM. This information includes the name and version of the SDEM. This section will contain other information related to the translation of the SDEM.

The final element of the side record is the name of the GML file used to map the side’s architecture/SDEM to the common representation used by the gateway. The

details on the GML files were described in the previous section.

The final section is filters. Some exercises use gateways to provide filtering that is not supported by architecture or to create enclaves. To support this, users have to have a way to specify filters. GCL provides an implementation independent format for defining filters. Each filter is given a name. In GCL a filter is specified for a single input side and multiple output sides. Each filter is defined only one way. GCL gives the user options to specify the filters. These include filtering based on entity ids, senders, class, and spatial. GCL provides additional control over each of these options.

GCL is not finalized at this time. Figure 4.1-1 provides an example of what GCL schema might look like.

5.2 GCL User Applicability

GCL helps our example user both in the near-term and in the long-term. In the near-term the user can use GCL to capture the required configuration information for both gateways. This capture includes the specialized filtering required for one gateway instance. Because GCL is implementation independent it is easier for members of the exercise to review and comment on the configuration of the gateways for the event. Making this information available for review to a broad audience can prevent problems later. This type of review is more difficult with the implementation specific configuration files. The same parameter may be named different in each implementation. In our example, users would have to be familiar with the formats of two different gateway implementations to perform the review.

GCL also helps our user in the long-term when it comes time to plan the next event. The user will have the GCL files from the previous event for reference. The user may have performed the gateway selection process again and decided to use different gateways from the previous events. GCL will make it easier to configure the new gateways similar to the previous event. The GCL files would become part of the documentation for the event.

5.3 Future Tools Based on GCL

GCL is a valuable tool for users as an independent format for documenting gateway configurations. Because GCL is based on XML, there are a number of free and commercial tools for creating and viewing XML files. However, some additional tools for using GCL would increase its value to users. Two of the potential tools are a tool for creating and viewing GCL and an API for reading GCL.

While many XML editors exist, many users are not comfortable using them. Therefore a specialized tool for creating GCL could be created. This tool would walk users through the creation process. In addition to providing support in creating the GCL file, the tool would help the user make decisions based on the available options. This tool could also be used by other members of the team to review the GCL files.

The GCL read API would provide a library for reading the GCL files. This could be used by gateway providers to create a converter that would generate their specific files. It could also be used by the developers to directly read the files into their gateways.

6. Future Efforts

The LVCAR Bridges and Gateways task is continuing. At this point GML and GCL are still in the initial concept phase. The languages will be finalized over the next year. It is possible that these languages will be proposed as community standards through SISO.

In addition to the languages, tools will be created to increase the value of the languages to the user. Some of these tools will be developed as part of this task.

7. Summary

This paper has discussed two of the products being developed by the LVAR Bridges and Gateways task. These products are intended to provide the user ways to better use gateways by documenting the configurations, and by using standard configurations to prevent being locked to a single gateway implementation. The work in this area is ongoing. There are plans to publish additional material as these concepts are further developed.

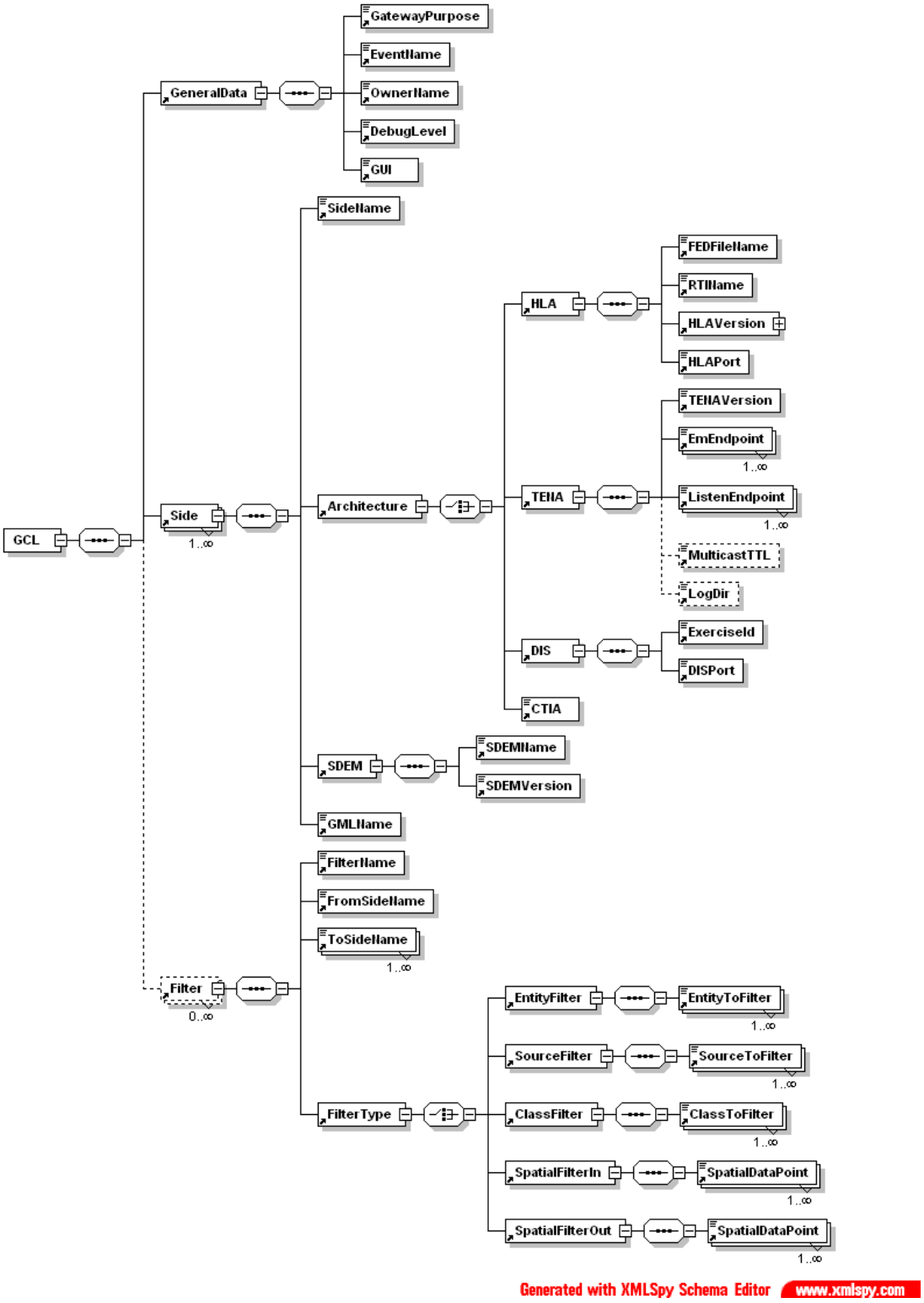

Figure 4.1-1: GCL Example Schema

8. References

[1] Live, Virtual, Constructive Architecture Roadmap (LVCAR) Final Report, Institute for Defense Analyses, September 2008.

[2] Drake D, Lutz R, Cutts D, Lessmann K, O'Connor, M, LVC Common Gateways and Bridges, 2010 I/ITSEC 10079, December 2010.

[3] Lutz R, Drake D, Gateway Concepts for Enhanced LVC Interoperability, 11S-SIW-024, Spring 2011 SIW, April 2011

[4] Lessmann K, Drake D, Cutts D, O'Connor M, LVCAR Enhancements for Selecting Gateways, 11S-SIW-54, Spring 2011 SIW, April 2011

[5] Wallace J, Bizub W, Ceranowicz A, Cutts D, Powell E, Gustavson P, Lutz R, Rheinsmith R, Bowers A, McCloud T, Object Model Composability and LVC interoperability Update, 09F-SIW-018, Fall 2011 SIW, September 2009

Author Biographies

MICHAEL O'CONNOR is a Senior Principal Engineer for ITT Corporation's Information Systems division. Mr. O'Connor has more than 20 years experience in modeling and simulation. He has been a key participant in the development of distributed modeling and simulation standards including IEEE 1278 and IEEE 1516. Mr. O'Connor was the editor for the SISO Real-time Platform Reference Federation Object Model (RPR FOM). 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 oversees the development of ITT's Chemical, Biological, Radiological, and Nuclear Simulation Suite, which supports DIS, HLA, and TENA. He has a Bachelors of Computer Engineering from Auburn University and a Master's of Science in Computer Science from the University of Alabama in Huntsville.

DANNIE E. CUTTS is a Senior Computer Scientist with AEgis Technologies Group Inc. supporting the USJFCOM Joint Advanced Training Technologies Laboratory (JATTL) in Suffolk, VA. He has over 20 years experience in Modeling and Simulation for NASA and the DOD and has been involved with the High Level Architecture (HLA) since 1995, serving on the Interface Specification and Time Management Working Groups. He has provided HLA Training and Cadre support for 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). He has supported numerous federation development efforts, as well as projects bringing legacy and new simulations to HLA Compliance. At USJFCOM he is involved in efforts to improve interoperability between Live, Virtual and Constructive assets for Joint Training.

KURT LESSMANN is a Vice President at Trideum Corporation, specializing in providing solutions and services in the area of test and evaluation, modeling and simulation, systems analysis, and information technology. Mr. Lessmann has over 15 years experience in supporting Live, Virtual, and Constructive (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 Test and Training Enabling Architecture (TENA) project and the Joint Mission Environment Test Capability (JMETC) Program Office as a senior Event Lead. Mr. Lessmann has a Bachelor of Aerospace Engineering from Auburn University.