Joint Composable Object Model and LVC Methodology

Richard Rheinsmith
Senior Enterprise Architect
Novonics Corporation
rrheinsmith@novonics.com

Jeffrey Wallace
Chief Technology Officer
Intelligent Integrations
jwallace@cots-llc.com

Warren Bizub
Director, Joint Advanced Concepts
Joint Warfighting Center, USJFCOM
warren.bizub@jfcom.mil

Dr. Andy Ceranowicz
Technical lead,
Joint Semi-Automated Forces (JSAF)
Alion

Dannie Cutts
Senior Computer Scientist,
AEgis Technologies Group Inc

Dr. Edward T. Powell
Lead Architect Test and Training Enabling Architecture
SAIC

Paul Gustavson
Chief Scientist
SimVentions, Inc

Robert Lutz
Principal Staff Scientist
Johns Hopkins University Applied Physics Laboratory

Terrell McCloud
Chief Architect Common
Training and Instrumentation Architecture (CTIA),
Lockheed Martin

Abstract. Within the Department of Defense, multiple architectures are created to serve and fulfill one or several specific service or mission related LVC training goals. Multiple Object Models exist across and within those architectures and it is there that those disparate object models are a major source of interoperability problems when developing and constructing the training scenarios. The two most commonly used architectures are; HLA and TENA, with DIS and CTIA following close behind in terms of the number of users. Although these multiple architectures can share and exchange data the underlying meta-models for runtime data exchange are quite different, requiring gateways/translators to bridge between the different object model representations; while the Department of Defense's use of gateways are generally effective in performing these functions, as the LVC environment increases so too does the cost and complexity of these gateways. Coupled with the wide range of different object models across the various user communities we increase the propensity for run time errors, increased programmer stop gap measures during coordinated exercises, or failure of the system as a whole due to unknown or unforeseen incompatibilities. The Joint Composable Object Model (JCOM) project was established under an M&S Steering Committee (MSSC)-sponsored effort with oversight and control placed under the Joint Forces Command J7 Advanced Concepts Program Directorate. The purpose of this paper is to address the initial and the current progress that has been made in the following areas; the Conceptual Model Development Format, the Common Object Model, the Architecture Neutral Data Exchange Model (ANDEM), and the association methodology to allow the re-use of multiple architecture object models and the development of the prototype persistent reusable library.

1. INTRODUCTION

One of the fundamental difficulties involved with mixed architecture live, virtual, and constructive environments is the coordination and correlation of the data exchange models that enable state sharing and interoperability. The Joint Composable Object Model (JCOM) project was chartered to address this problem, and its progress to date is described in this paper. The principal results are the design of an eight phase process for data exchange model composition, and the creation and integration of the infrastructure required for its implementation. This paper will cover: the JCOM concept of operation including the composition process, application of conceptual modeling, the Architecture Neutral Data Exchange Model (ANDEM), and a discussion of the enabling metadata.

A quick detour into terminology is needed at this point to identify and define the key terms used in this paper, as they are interpreted broadly in the community. 'Data exchange model' (DEM) refers to the structure of the data used to communicate state and state changes between cooperating simulations. We use DEM instead of the more common term 'Object Model', such as the Federation Object Model (FOM) used by the HLA, to avoid confusion with software object models which include functional aspects. The term 'object' in distributed simulation originates from the fact that many messages used in a DEM are updates for the state of a real or simulated object such as a person or vehicle.

As such, the term 'object' is used in LVC environments in the common sense rather than the software sense. Heavy use is made of the term 'component' in the general sense, indicating units that can be composed to create larger units, essentially reusable piece parts. DEMs are primarily composed of messages, and messages are composed of attributes and all of these are components of a DEM. Simulations are components of LVC federations. The LVC and distributed simulation community often refers to messages as classes and allows the use of inheritance to extend messages. We use 'conceptual models' to refer to abstractions of real or synthetic worlds that we want to include in our LVC environment. These abstractions include entities, processes, events, and states. 'Model' and 'representation' are used as equivalent terms; thus 'data exchange model' is equivalent to 'data exchange representation'.

In order to reuse DEMs efficiently, an easy way to find and retrieve them is necessary. An intelligent, searchable repository for DEMs must be built; allowing many new DEMs to be composed from existing ones. This should be a repository rather than a registry, because for efficiency the engineer should be able to retrieve the DEMs that match his search criteria immediately as opposed to a registry that tells him who to call to get the DEM.

This repository needs to contain the links between conceptual models of the domain and data exchange model components. Standard repository development techniques employing simplistic metadata descriptions are not sufficient to support semantic, concept-based queries. While the project intends to improve conventional metadata description initially, for the long term it will rely on

the open standards, methods, and technologies that have been developed for application areas such as the Semantic Web[1] to support semantically rich repositories and queries.

The essence of the JCOM project is to show how conceptual models of the domain can be used to organize and select data exchange model components which can be rapidly composed to create new LVC environments for training, experimentation, and other purposes. While this approach can be used to augment current federation building processes, only by leveraging semantic technologies can long-term breakthroughs in speed and accuracy of composition be achieved.

The basic JCOM concept of operation is for existing object models from the different LVC interoperability architectures to be parsed into an architecture neutral data exchange model format and stored in a repository.

2. COMPOSITION PROCESS

In this compositional development environment, LVC federation creation may be viewed as a constructive activity. A simulation of the desired functionality is composed from a set of existing LVC components. The LVC components are interfaced together via DEM components and the composition process produces a composite DEM that can connect all the LVC components required to implement the desired composite LVC federation.

In this Compositional Model of DEM development, the Accumulation, Evaluation, and Adaptation activities can be conceptually grouped into the process of Reuse. Feedback occurs between the Conceptualization and Reuse processes when conceptualization is influenced by the availability of components. This influence can be either in the form of repartitioning within the parameters of the original design, or of relaxing design constraints. If no candidate artifacts are found to satisfy the requirements, the designer may revise the conceptualization under a different design strategy to increase the opportunity for reuse, or may elect to implement the needed component (Prieto-Diaz 1987).

Feedback also occurs between the Reuse and Composition step when interface requirements dictate certain adaptations that may not be feasible with a particular artifact.

Standard development methodologies fail to support the compositional development model in three important ways. The compositional development processes of Accumulation and Evaluation are most tractable when object model definitions are independent, but this is often not the case. Most data and object modeling approaches lack support for representing the inter-object relationships that can capture this dependence. They only support two kinds of inter-object references, inheritance (IS-A) and client (HAS-A) relationships. From the standpoint of reuse, this is insufficient, because coupled components cannot be evaluated independently and the accumulation and evaluation processes take on a combinatorial aspect.

The second problem involves methodologies based on class reuse. Class-level reuse often occurs at too fine a granularity to be effective. It has been noted by other researchers that the advantages involved in reusing a component increase super-linearly as the component grows in size (Biggerstaff 1987). Thus a methodology that allows the reuse of larger components is more effective.

The third criticism of reuse support observes that object-oriented design methodologies only offer the developer *syntactic* support and only after the conceptualization, accumulation, and evaluation process has produced a candidate object for adaptation. Object-oriented methodologies offer this support through inheritance allowing the developer to "design by difference," adapting a chosen component through inheriting the candidate object into another class and specializing its structure. However, there is considerable intellectual challenge in the compositional processes of conceptualization, accumulation, and evaluation which need support.

This type of inheritance makes object and data model maintenance and evolution harder because the inheritance relationships violate the semantic model of the system. In recognition of the fact that object-level approaches are inherently insufficient to facilitate large-scale improvements in reuse, researchers have begun to look at higher-level abstractions and compositions; in the object-oriented community, these abstractions are referred to as design patterns and frameworks (Johnson 1988, Gamma et al. 1994, Whitehurst 1997), while non-object oriented systems research refers to these abstractions as reusable architectures.

[1] http://www.w3.org/2001/sw/

To begin development some method is needed to capture a conceptual model description that represents the training objectives in a format that can be algorithmically processed to support discovery and selection. The Joint Capability Areas (JCA) and Mission Essential Task List (METL) are good resources for building these conceptual models. Discovery and selection require that the DEM components (DEMCs) represented in ANDEM and stored in the repository are indexed by the same conceptual models that are employed to describe the training or experimentation tasks. A search mechanism that is capable of utilizing the conceptual model's semantically rich metadata developed in Phase One is required to match up the training objectives to the DEMCs.

Composition requires the ability to quickly and easily manipulate the inheritance and composition relationships of and between DEMCs is important. The ability to merge the graph structure representation underlying the DEMC is required, in addition to the ability to join, and potentially re-label the schemas of the DEMCs. This activity and capability is at the core of object and data exchange model composition.

Implementation of new OMCs/DEMs requires the ability to create new artifacts. As such, authoring and editing tools such as those commonly found in standard data modeling or object modeling tool environments would be desirable. The issue is the integration of such a capability into this composition environment and process. Typically such tools are stand alone and have limited import and export capabilities that permit the interoperation with other tools and processes. The principle capability required for the expansion process is the ability to quickly and easily send new and adapted DEMCs back into the repository to fulfill future task or mission requirements. The final phase of adding architecture specific information requires the ability to manipulate and augment the ANDEM data structures in a flexible manner.

The notion of an end-to-end Integrated Development Environment (IDE) for an object model composition process needs to be developed. Support tools, such as Protégé, GraphML, and Xerlin (for XML editing) should be combined through open source IDEs, which provide the necessary flexibility, through plug in creation and implementation.

3. CONCEPTUAL MODELING

What this means for the war-fighter: Rapid and efficient federated simulation development. Current technologies require considerable time to create a complex multi-architecture training and experimentation environment. As a result, a few established federated LVC environments are relied upon, where users are forced to make do with what exists, which means their requirements are not necessarily met. Conceptual modeling has been found to be a key part of the Object Model composition process. Conceptual modeling describes what is to be represented, the assumptions limiting those representations, and other capabilities needed to satisfy the user's requirements (IEEE P1730). In general, the conceptual model must identify the distinct entities or phenomena involved in the mission thread under consideration. It must also identify the actions of entities and the collaborative actions or activities that take place between them. These actors and common behavioral patterns are captured in a machine understandable form capable of triggering a semantic search.

Without a structured method for conceptual modeling, automating, or even semi-automating, the process of mapping between training and experimental objectives and the DEMs supporting them is challenging. As such, the ad-hoc processes for building federations will continue. In addition, the problem of finding and integrating LVC environment resources is made more difficult by the presence of multiple LVC integration architectures. There are frequently separate assets, subject matter experts (SMEs), DEMs, and repositories. Conceptual models are necessary to organize all these resources under a uniform schema that allows reuse independent of interoperability affiliation. Conceptual modeling can also help the LVC community to move away from the specialized terminology of M&S to that of the War fighter and live ranges. This will make M&S more understandable and useful to the War fighter.

Typically upfront conceptual modeling and analysis is limited and sometimes non-existent. Defining the scope of a project; understanding requirements and the way forward – pairing with what is needed to what is to be built and used is critical. Projects are often limited in applying conceptual modeling because of budget/personnel/resource constraints. Another obstacle is that using Object Model design for discussing capabilities with stakeholders may be "too big of a leap". Without knowing where to go "to mine" (defining / integrating) reusable

conceptual models each project is left with the overwhelming task of defining everything from scratch.. Contracts rarely include contractual obligations to support Conceptual Model development, delivery and reuse. All these problems can be helped with the creation of a structured methodology for reusable conceptual modeling and sharing conceptual models can make better use of limited resources for conceptual model development. In general, Conceptual Modeling needs to be emphasized more fundamentally as an activity that not only assists in implementation, but also helps programmatic judgment.

In the initial phase of the JCOM project techniques such as the Object Modeling Groups (OMGs) Unified Modeling Language (UML) were employed to represent the conceptual models of a sampling of authoritative mission threads that could realistically be required as a part of a training exercise, experiment, or test and evaluation event. Additionally the Base Object Model (BOM) template specification (which employs UML sequence and activity diagrams) has been studied as an example of conceptual modeling. Some of the questions considered are:

- How do we extract the "piece parts" of existing object models that correspond to conceptual model components?
- How do we define the mapping from a conceptual model component to a corresponding object model component?
- How do we compose whole object models from a set of object model components?

Conceptual models for the JCOM effort have been captured using sequence diagrams The sequence diagrams provide the opportunity to identify common patterns, where these patterns can be extracted and potentially reused. Using the sequence diagram an entire mission thread can be examined and understood at the high level. The sequence can then be reviewed and decomposed to further explore the layers of sub-patterns that compose the mission thread. As the mission thread is further decomposed, the patterns may expose more details and variations such breadth of entity types (e.g. HQ at the mission thread layer includes Division, Brigade, and Company at the lower sub-pattern layers).

In addition to capturing the patterns of interplay, the conceptual model also identify types of conceptual entities required and their states providing a means to understand entity behavior that would need to be represented by a system or simulation. For example, in our original pattern of interplay, three conceptual entities were identified: Target, Observer, and HQ. For the Observer, there are three states associated to this entity: Observe, Decide, and Communicate. These are states are reflected in the figure above.

4. ARCHITECTURE NEUTRAL DATA EXCHANGE MODEL

What this means for the war-fighter: The effective and efficient reusing of multiple architecture products regardless of service, component, or development tool. The independent format allows mapping any interoperability architecture DEM to a common language. Once mapped, it will support reuse in multiple interoperability environments.

The question is not whether one object model can be mapped to another. The use of gateways to bridge the multiple LVC architectures is prima facie evidence that architecture specific DEMs can be mapped to each other. That problem is solvable by developers familiar with the models involved. The problem at hand is to accelerate and automate as much of the mapping process as possible. There is strong agreement that an Architecture Neutral Data Exchange Model (ANDEM) format for data exchange models can simplify the problem both for humans and machines. Humans can handle the problem for specific federations since the number of OMs that need to be translated between in a particular exercise environment is small. However, once the general problem is attempted, the large number of potential OMs necessitates a many to one approach rather than a many to many approach.

To create the ANDEM, JCOM started with the goal of extracting a single data exchange metamodel from the metamodels for TENA, HLA, DIS, and CTIA. This metamodel should be able to express the same data exchange capabilities as any TENA, HLA, or DIS object model. In the process there was disagreement as to whether ANDEM should be the intersection or union of these architecture specific metamodels. The intersection produces abstraction which is necessary for recognizing equivalence between different data exchange. For example, if transmission reliability were a necessary parameter of ANDEM, then there would never be equivalence between any HLA FOM component that uses reliable data transfer and the DIS Protocol Data Units – even though they may describe exactly the same world state.

However, once equivalence between two data exchange models has been established, there is the requirement for synthesis and implementation, which cannot be automated without capturing the specific implementation options of each protocol. Thus it was decided that in addition to the ANDEM, an architecture specific extension, or appendix, would need to be kept for each data model for use in building the translation between the formats. Yet, even then it was not easy to separate the conceptual pieces from the implementation pieces.

architectures and are orthogonal to the common core.

Figure 2 depicts the Architecture Neutral Data Exchange Model (ANDEM) core metamodel. As stated previously the goal is to create a structure into which all of the constructs present in the four major LVC architectures can map into. This includes the three variations of HLA, the 1.3NG,

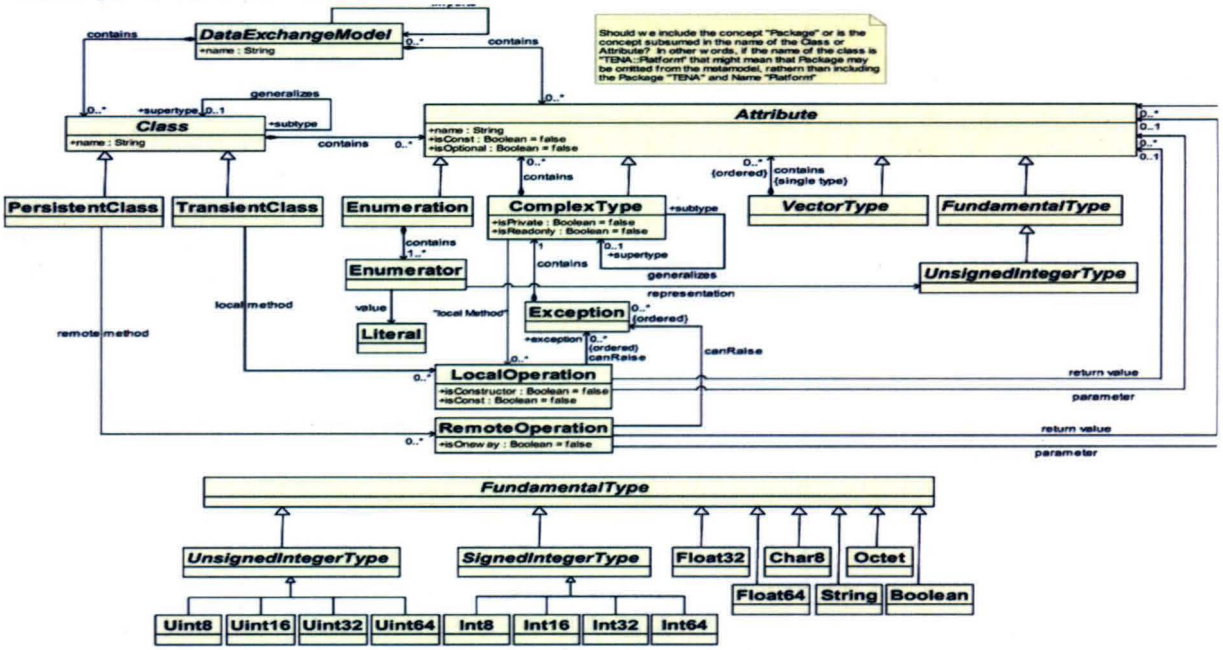

Figure 2. ANDEM Metamodel Prototype

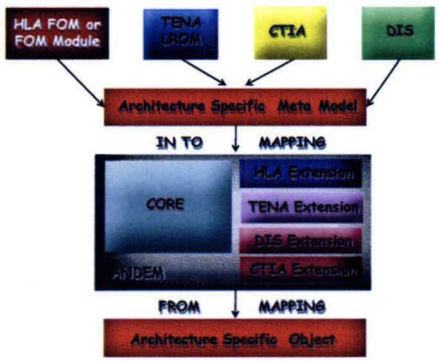

Figure 1. ANDEM Concept of Operation

Figure 1 illustrates the concept of operation of the Architecture Neutral Data Exchange Model (ANDEM). The ANDEM core represents the constructs that are common between each of the LVC architectures plus those constructs that materially affect a useful metamodel structure. This means including constructs that are not shared by all of the LVC architectures. The ANDEM architecture specific extensions represent those constructs that are unique to one of the LVC

IEEE 1516, and HLA Evolved. For example, the current ANDEM includes primitive data types, which lies in the intersection of all our three prototypical data exchange metamodels. Another question that arose was whether Live Architecture / Data models are adequately represented in the current set of the four LVC architectures under consideration.

The central feature of the ANDEM metamodel is the notion of a Class, which is the fundamental unit of representation. This concept exists in all four LVC architectures. The notion of inheritance is also present, even though it is not strictly present in all four, as is composition by inclusion (HAS-A relationships). The data exchange model being in several separate files is a construct that presently exists only in HLA Evolved and TENA. The notion is central enough that it is included in the ANDEM core metamodel, as such a construct would be difficult to retrofit.

The class construct has two sub-types, the persistent class and the transient class. The

distinction is made between classes that represent entities whose state persists over time (e.g., a platform or a sensor) and those that do not, such as weapon firing events or communication. The main feature of both types of classes is the ability to contain other classes or an attribute. As is indicated in the figure above the attribute construct has four variations: Enumerations, fundamental type, Vector type, and Complex types

The specialization of the fundamental type is standard across all of the LVC architectures. Notice that the vector type is configured to accommodate a single type of any attribute. Strictly speaking, the construct is not present in all of the architectures but its inclusion here is most natural.

5. METADATA AND COMPOSABILITY SERVICES What this means for the war-fighter: A simple but robust method for categorizing everything from a handgun to the newest air superiority jet.

Making previously created artifacts easy to find and retrievable should help alleviate reimplementation due to the common expedient of "I can't find it so I'll just create a new one". In conjunction with a structured conceptual model and rapid reuse of multiple architectures in the LVC community, this technology will allow commanders at all levels to better understand and apply their tools.

There is agreement that ontologies as metadata, and related tools to create and maintain them offer great promise for the future in terms of composability support. Ontologies enable reduction in ambiguity of specification, and will reduce the current labor intensive processes required to create data exchange models. They will also permit and facilitate archiving and maintaining interoperability knowledge that is typically lost, or kept only by original designers.

6. SUMMARY

This papers summarizes the JCOM project, along with the strategy and supporting technologies needed to achieve those goals. JCOM is considered just the first step in a longer and more extensive process to promote convergence and improve LVC interoperability. While object modeling is just one aspect of the broader LVC interoperability problem, the products and lessons learned from this project will provide a solid foundation for follow-on initiatives.

Questions or comments related to the conduct of this effort may be directed to the JCOM Program Manager, Mr. Warren Bizub (warren.bizub@jfcom.mil).

ACKNOWLEDGEMENTS

The authors thank Barbara and Jaclyn Hannibal for their support in the editing, graphics arts, and production support of this paper.

REFERENCES

Biggerstaff, Ted J. and Charles Richter. *Reusability framework, assessment and directions*. IEEE Software, 4(3):41-49, March 1987.

Daniel Elenius, et al, 2007. "Purpose-Aware Reasoning about Interoperability of Heterogeneous Training Systems," In the *Proceedings of the 2007 International Semantic Web Conference*, Semantic Web Science Association, Busan, Korea, August 20, 2007

Gamma, Erich, et al., 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.

IEEE P1730, "Distributed Simulation Engineering and Execution Process", *IEEE Draft Recommended Practice (V3)*, 2009.

Johnson, Ralph and Brian Foote. *Designing reusable classes*. Journal of Object-Oriented Programming, 1(2):22-35, June 1988.

Prieto-Diaz, Reuben and Peter Freeman. *Classifying software for reusability*. IEEE Software, 4(1):6-16, January 1987.

Sowa, John, 2000. *Knowledge representation: logical, philosophical, and computational foundations*, MIT Press.

Tolk, Andreas et al., 2008. "Implied Ontological Representation within the Levels of Conceptual Interoperability Model," *Intelligent Decision Technologies,* IOS Press, Vol 2, Number 1, 2008.

Wallace, Jeffrey and Barbara Hannibal, 2008. "Software and Hardware System Integration and Intelligent Automation using Ontology-based Knowledge Representation Technology," In the *Proceedings of the 2008 International Conference on Artificial Intelligence*, World Academy of Sciences, Las Vegas, NV, July 14-16.

Whitehurst, R. Alan, 1997. "Using IMPORT to Implement Complex Behaviors in Simulations," In the *Proceedings of the 1997 Object-Oriented Simulation Conference*, The Society for Computer Simulation, Phoenix, AZ, January 12-15, 1997.