# Object model composability and multi-architecture LVC interoperability

# Object Model Composability and Multi-Architecture LVC Interoperability

**Jeffrey Wallace**
**Andy Ceranowicz**
**Ed Powell**
**Robert Lutz**
**Andrew Bowers**

**Warren Bizub**
**Dannie Cutts**
**Paul Gustavson**
**Richard Rheinsmith**
**Terrell McCloud**

## ABSTRACT

The development and execution of mixed-architecture Live, Virtual, and Constructive (LVC) environments poses many challenges (e.g., employing some combination of DIS, HLA, TENA, and CTIA). The LVC Architecture Roadmap (LVCAR) initiative was established to document these challenges. One key finding addressed the need to gain cross-community agreement on the format and content of object model data. Further, LVCAR recommended that agreements on object model content occur at the atomic-level and tools and techniques be developed to support the assembly of object models from object model components. The purpose of this paper is to report the progress of a pathfinder project to address this problem, called the Joint Composable Object Model (JCOM), which is developing an object model composability approach.

There are three aspects to the proposed solution: First, a technical framework to formalize the LVC environment conceptual model to relate to and support warfighter missions. The formalization of conceptual modeling provides a method to define LVC system functionality in a way to enable object model component composition via mission threads, organizing the conceptual model content while providing relevancy to warfighters. Second, the paper presents an architecture independent object model format as a method and technology to enable effective composability. This is critical to enable mixed-architecture LVC environments, permitting the reuse of existing assets. Third, since agreements on object model content and format are easier to achieve at a finer granularity, the approach relates object model fragments to the conceptual model via descriptive metadata. Metadata descriptions provide the method and semantic content required to relate the various object model components, and support object model assembly. Example mission threads, including Joint Close Air Support (JCAS), Time Sensitive Targeting (TST), and Consequence Management, showcase the methodology.

## ABOUT THE AUTHORS

**Jeffrey Wallace** is the Chief Technology Officer of Carpe Occasio Technology Systems (COTS). He has an M.S. and B.S. in Mathematics, University of Illinois Urbana-Champaign, and has over 25 years experience in unmanned systems, interoperability, artificial intelligence, modeling and simulation (M&S), and high performance computing - in a variety of academic, government, and industry positions.

**Warren Bizub** is the Director, Joint Advanced Concepts for the Joint Warfighting Center, USJFCOM. He has also served as the Technical Management Division Chief of the Joint Training & Education Capability Group and Director, Advanced Training Technologies Group while at USJFCOM.

**Dr. Andy Ceranowicz** is a Chief Scientist at Alion and technical lead for federation and Joint Semi-Automated Forces (JSAF) development at JFCOM J9. He led the development of the Millennium Challenge 02 federation as well as the development of JSAF and its predecessors, ModSAF and SIMNET SAF. Andy holds a Ph.D. in Electrical Engineering from The Ohio State University.

**Dannie Cutts** is a Senior Computer Scientist with AEgis Technologies Group Inc. supporting the U.S. Joint Forces Command (USJFCOM) Technical Development & Innovation (TD&I) branch in Suffolk, VA. He has over 20 years experience in Modeling and Simulation (M&S) 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.

**Dr. Edward T. Powell** is a Lead Architect from SAIC for the Test and Training Enabling Architecture. After receiving his Ph.D. in Astrophysics from Princeton University, he worked for the Lawrence Livermore National Laboratory performing simulation-based analysis. Ed participated as lead architect in some of the most complex distributed simulation programs in DoD, including the Joint Precision Strike Demonstration (JPSD), the Synthetic Theater of War (STOW) and Joint Simulation System (JSIMS).

**Paul Gustavson** is the chief scientist at SimVentions, Inc, and has more than 19 years of experience, including the design and development of DoD standards, simulation systems, software applications, and numerous technical publications and tutorials on simulations and software development. He is an active leader within the Simulation Interoperability Standards Organization (SISO), serving as product lead and architect of the Base Object Model (BOM) concept, and is involved in other standards efforts, including the Distributed Simulation Engineering and Execution Process (DSEEP).

**Robert Lutz** is a principal staff scientist at The Johns Hopkins University Applied Physics Laboratory in Laurel MD. He has over 28 years of experience in the design, implementation, and evaluation of modeling and simulation (M&S) systems for military customers. Currently, he is leading several M&S standards initiatives within the Simulation Interoperability Standards Organization (SISO), including the Simulation Reference Markup Language (SRML) standard, the OMT component to the IEEE 1516 HLA standard, and the IEEE 1730 Distributed Simulation Engineering and Execution Process (DSEEP) standard.

**Richard Rheinsmith** is a Senior Enterprise Architect with the Novonics Corporation, and a 23 year Army veteran with extensive experience in Training, Education, and Engineering. Since his retirement in 2000 he has worked extensively in the modeling and system architecture arena for the Army, Navy, and Joint Forces Command. His current position is providing support to the Joint Forces Command J7's Advance Concepts.

**Andy Bowers** is a Modeling and Simulation Engineer with MITRE supporting the U.S. Joint Forces Command (USJFCOM) Technical Development & Innovation (TD&I) branch in Suffolk, VA. A retired U.S. Army officer, Mr. Bowers has been involved in simulation-supported training for over fifteen years. Mr. Bowers has supported several federation development efforts and is currently the FOM manager for the JMRM and Marine Corps federations.

**Terrell McCloud** is a senior staff software systems architect at Lockheed Martin Simulation, Training and Support. He is currently the Architect of the PEO STRI Common Training Instrumentation Architecture (CTIA) program, helping the Army transform its live training systems. His professional experience includes 15 years of developing software intensive systems with 10 years of Live Training modeling, simulation and integration. His interests include leveraging service oriented architecture technologies in live, virtual, and constructive training environments. He holds a B.S. in Electrical Engineering from the University of Central Florida.

# Object Model Composability and Multi-Architecture LVC Interoperability

**Jeffrey Wallace**
**Andy Ceranowicz**
**Ed Powell**
**Robert Lutz**
**Andy Bowers**

**Warren Bizub**
**Dannie Cutts**
**Paul Gustavson**
**Richard Rheinsmith**
**Terrell McCloud**

## 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. A current weakness of both repositories and registries is the effectiveness of their metadata in helping to find the best DEM components for a particular exercise or experiment. Accordingly, a new repository infrastructure is required with a foundation based on a semantically rich conceptual representation of the entities, tasks, or missions that the DEM components support.

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

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

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.

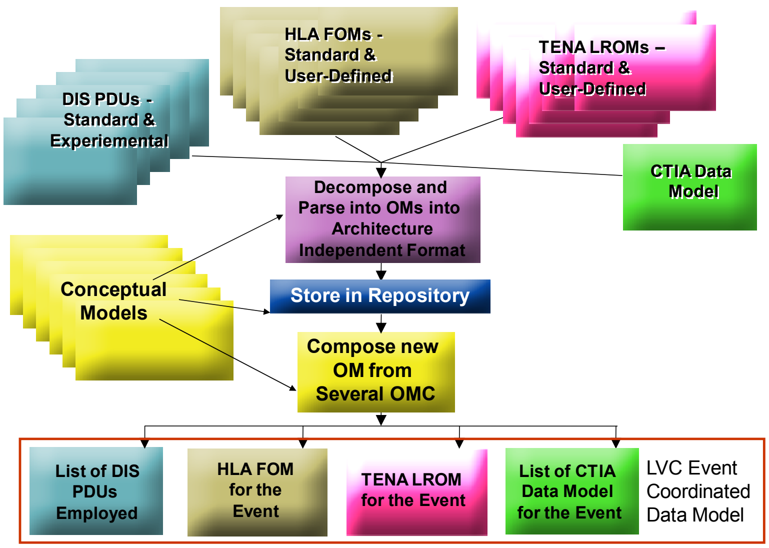


**Figure 1. JCOM Concept of Operation**

Figure 1 illustrates the basic JCOM concept of operation. Existing object models from the different LVC interoperability architectures are parsed into an architecture neutral data exchange model format and stored in a repository. The process shown in Figure 2 is employed to assemble the correlated data exchange model artifacts. The process produces a coordinated set of event/exercise specific DEM components that supports the tasks and missions identified in the conceptual model. In addition to the DEM it is also necessary have access to the LVC environment components behind the DEM components. For the purposes of this project the focus is on the DEM and assumes that each DEM component has an appropriate pointer to suitable underlying LVC components which are easily accessible to the LVC system developer.

## COMPOSITION PROCESS

In this compositional development environment, illustrated in Figure 2, LVC federation creation may be viewed as a constructive activity wherein 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.

When building systems from components, development can be characterized by a number of processes, including:

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).

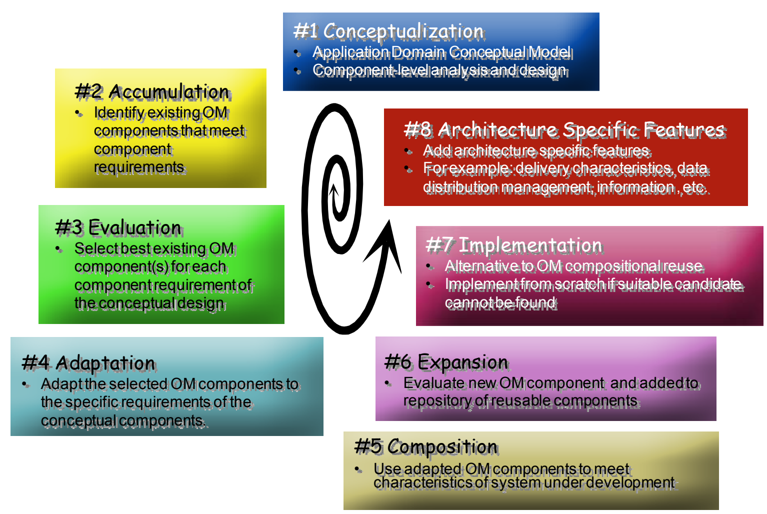


**Figure 2. JCOM Composition Process**

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. Even this limited syntactic support has its drawbacks:

1. Object and data structures created using excessive inheritance are harder to debug, because an object class's true structure is dependent upon the structure of its entire chain of ancestors, whose definitions may be scattered about the entire object or data model (and in different files depending on the LVC architecture employed – for example TENA or HLA Evolved); and

2. This form of reuse can lead to "spaghetti inheritance," where objects are linked through inheritance simply because they share a set of common features, even though they may be conceptually/semantically unrelated.

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. In general, composability is best facilitated using HAS-A relationships as opposed to IS-A[2], so it is important that composability tools include support for the HAS-A relationship.

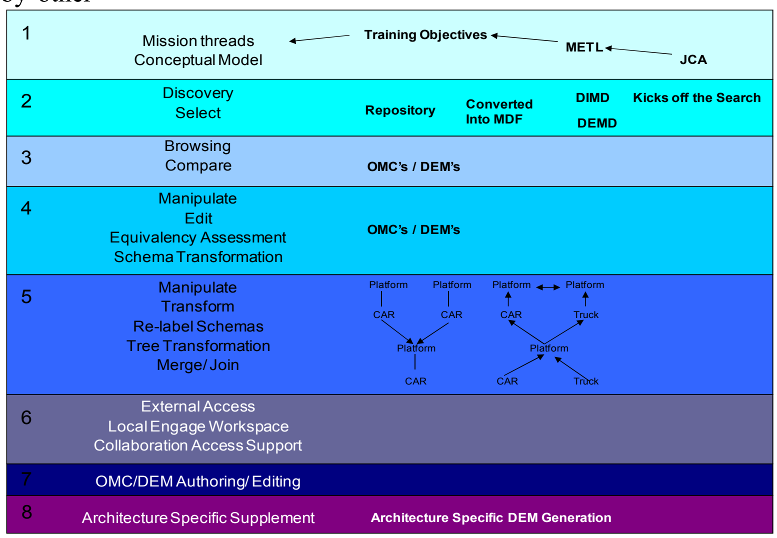


**Figure 3. Tool Support Requirements**

Figure 3 illustrates the requirements necessary to implement the composition process. To begin, 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.

---

2 http://www.artima.com/lejava/articles/designprinciples4.html

An example of a system capable of performing this kind of indexing and search is the Open Net-centric Interoperability Standards for Training and Testing (ONISTT) (Elenius, et al. 2007). ONISTT has an ontology framework that represents missions and tasks, along with interoperability and functional constraints, into which the machine readable conceptual model is mapped. Next, ONISTT has a resource ontology framework which is used to represent the components available to satisfy the task or mission requirements. The ONISTT resource ontology framework is an example of semantically rich metadata that describe the relationships between components and their capabilities. The ONISTT system analyzer is capable of selecting the components capable of satisfying task and mission requirements.
Such a selection process is likely to retrieve several candidates for each component.

Evaluating the candidate DEMC requires the ability to browse and compare the object model components or data exchange models, and create and maintain a collection in some sort of workspace or project format. The tools must be able to quickly and easily visualize and compare the DEMCs, in addition to evaluating the differences between the components from both a syntactic and semantic perspective. Adapting the DEMCs requires the ability to manipulate and edit them. Ideally the ability to transform schemas would be available to accommodate the case where a DEMC semantically fulfills a requirement but the data structure needs to be altered in order to meet syntactic needs.

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. This has already been demonstrated by the TENA Community through their use of the open source IDE Eclipse.

## 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. If the process of creating new federations was accelerated, many new customized and more efficient LVC environments could be established to address new and unique training and experimentation requirements that require fewer resources to operate.

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 Warfighter and live ranges. This will make M&S more understandable and useful to the Warfighter.

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 "pieceparts" 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?

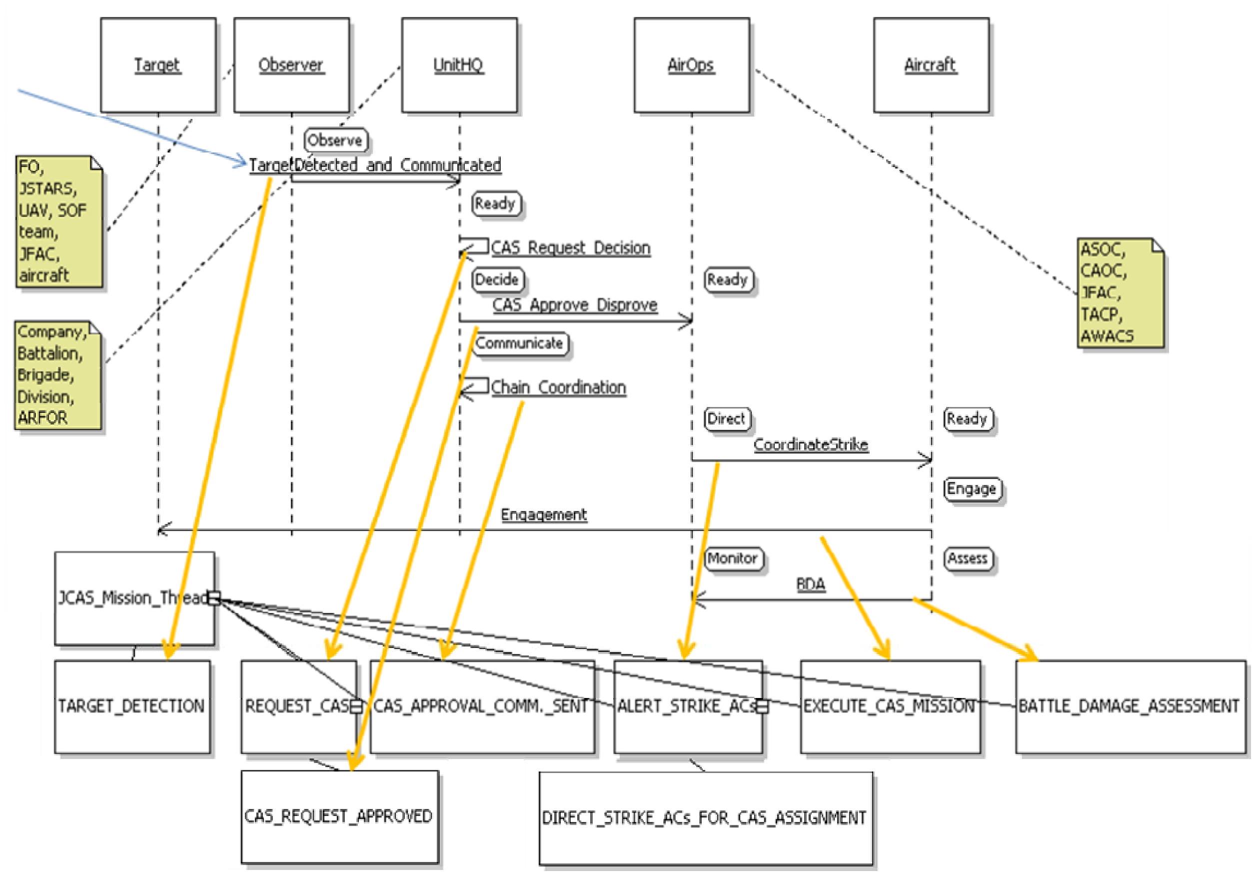


**Figure 4. JCAS Sequence Diagram**

Conceptual models for the JCOM effort have been captured using sequence diagrams, referred to as patterns of interplay in BOM terminology. The sequence diagram in Figure 4 portrays a common pattern pertaining to target detection. In this example a Target's signature is exposed to an Observer, who then determines if it is a threat, and if necessary requests Call For Fire from Headquarters. This pattern, which is captured using the conceptual model component of a BOM is actually a subcomponent of a larger mission thread known as Joint Combat Air Support (JCAS).

In this view, the entire mission thread can be examined and understood at the high level, 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, 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 below.

The SISO BOM technology for conceptual modeling was developed at a time when it was anticipated that HLA was going to be the only interoperability architecture. Thus the mapping from concepts to DEMS focused on HLA FOMs. The JCOM project has been working on generalizing this linkage to utilize our architecture independent DEM representation. This will allow BOM technology to map conceptual models to DEMs from any interoperability architectures. The JCOM project is also exploring how to take advantage of the more powerful representation and matching technology provided by ONISTT for conceptual model representation and matching, such as the ONISTT query method provided by the analyze function aspects.

## 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.

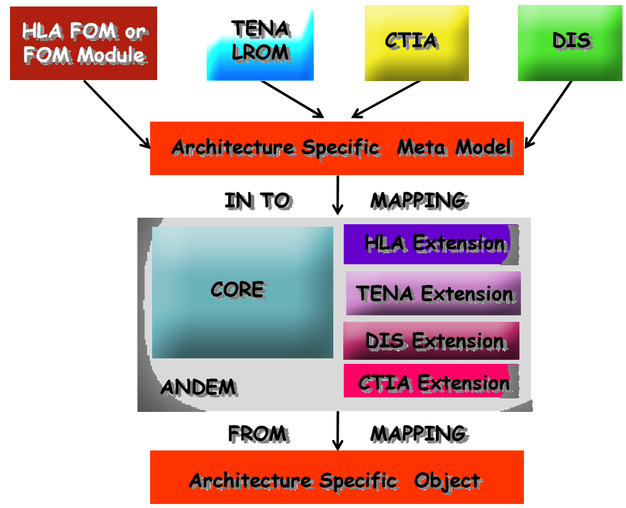


**Figure 5. ANDEM Concept of Operation**

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.

Figure 5 illustrates the concept of operation of the Architecture Neutral Data Exchange Model (ANDEM). As is indicated, an architecture specific metamodel exists for each of the LVC architectures, and a mapping into ANDEM and from ANDEM enables the JCOM CONOPS. 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. As will be seen subsequently, 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 architectures and are orthogonal to the common core. For example, information that describes delivery quality of service or data distribution characteristics fall into the category of extension because they are not shared by all the LVC architectures and do not materially affect the core structure by their inclusion.

Figure 6 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, 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, and the suggestion was made to add the JC3IEDM metamodel to our sources. This has not been done yet.

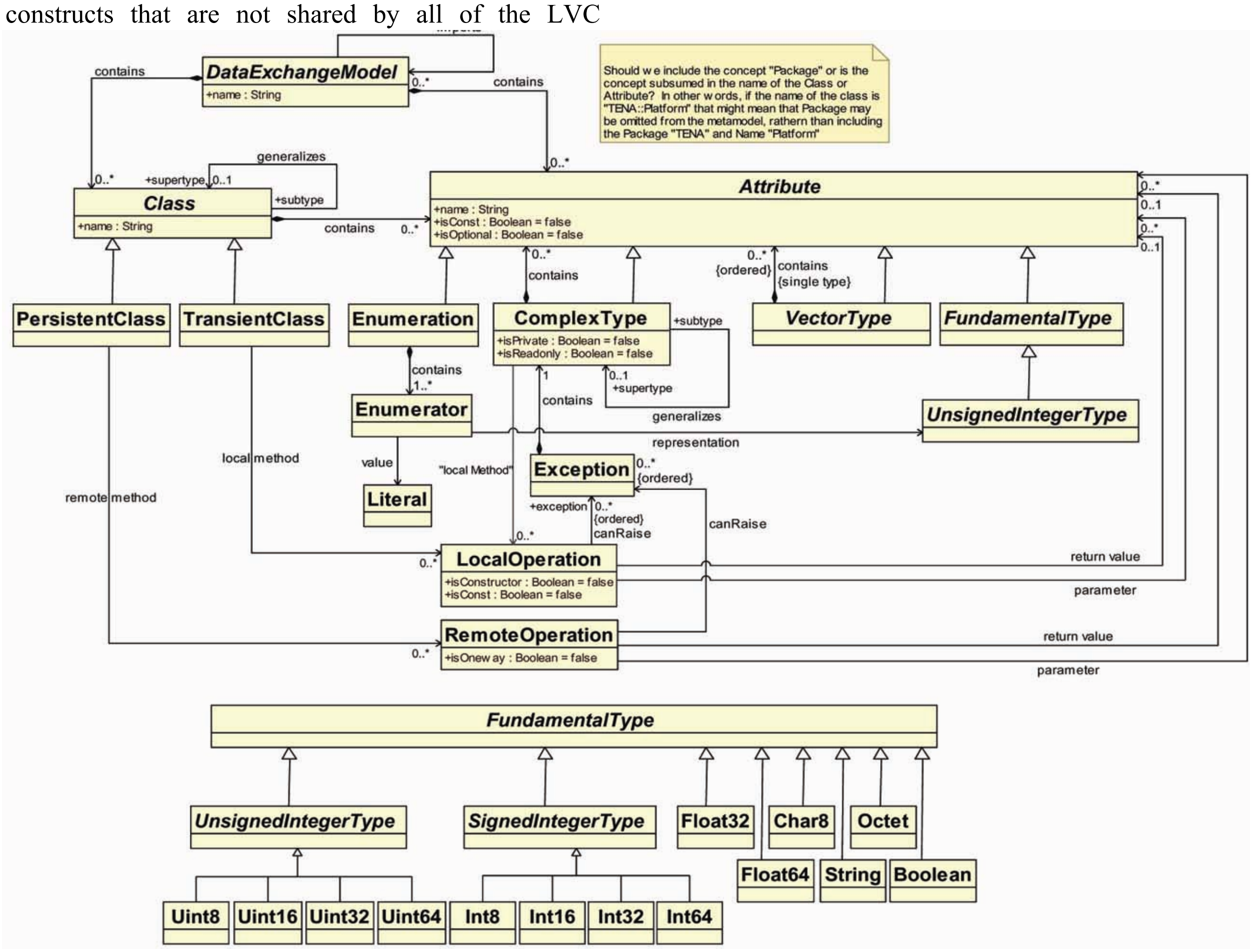


**Figure 6. ANDEM Metamodel Prototype**

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 into an architecture specific extension.

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
- 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. In a similar manner the local operation and remote operation constructs are represented in the ANDEM core even though this supports only TENA. The rationale is that incorporating the construct in an extension or supplement would be inelegant and difficult to implement in that manner.

## 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 retrieve 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

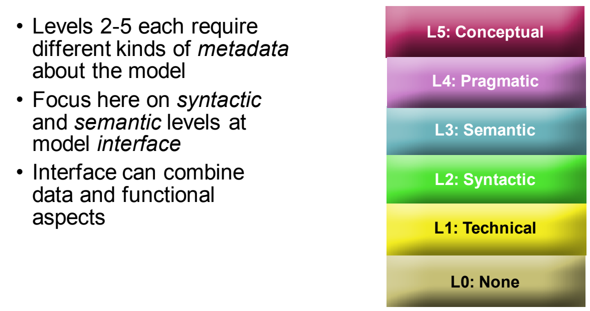


**Figure 7. Interoperability Levels**

One of the key points is that different types of metadata are required to support various levels of automation and interoperability. For example, Figure 7 (Tolk 2008) introduces the notion of levels of interoperability. The different levels require different structure in order to represent phenomenology at each level. For example, the technical and syntactical are the networks and protocols by which cooperating simulations communicate. The semantic level can be supported by object models, and the pragmatic level is concerned with state changes of the system based on the information exchange, and representations such as DEVS are useful in that regard as are the patterns of interplay defined in the BOM. The conceptual level can be addressed by conceptual graphs (Sowa 2000, Wallace 2008) and ONISTT. This addresses everything that is not implementation, and represents the metadata the JCOM project is concerned with, namely the ability to represent capabilities, assumptions, and constraints.

Figure 8 describes the infrastructure that supports the semantic metadata approach. The idea of a 'semantic web' built on markings expressing semantic content instead of the simple formatting provided by HTML. Furthermore, it is not possible for a machine to appropriately interpret meaning from just the order relationships of letters and words such as provided by XML. XML provides an elemental syntax for content structure within documents, yet associates no semantics with the meaning of the content contained within. XML Schema is a language for providing and restricting the structure and content of elements contained within XML documents.

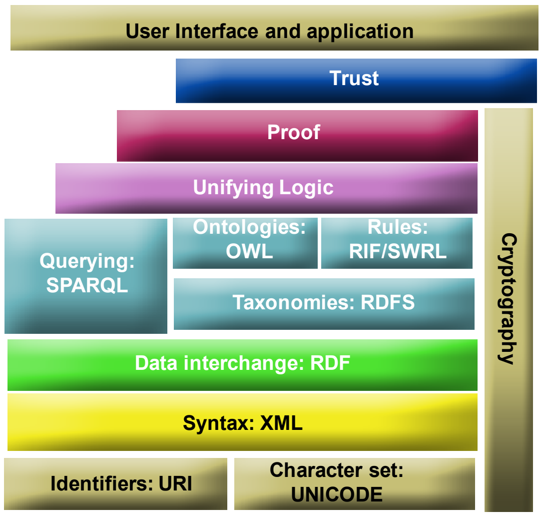


**Figure 8. Semantic Web and Metadata Architecture**

RDF is a simple language for expressing data models, which refer to objects ("resources") and their relationships. An RDF-based model can be represented in XML syntax. RDF Schema is a vocabulary for describing properties and classes of RDF-based resources, with semantics for generalized-hierarchies of such properties and classes. If everyone had their own RDF schemas there would be little semantic interoperability, but many researchers are now building publicly available schemas on the web that multiple designers can reuse thus establishing common vocabularies. RDF uses Universal Resource Indentiers (URIs) as its data elements to achieve uniqueness on the web and avoid ambiguity which machines don't deal with well.

OWL adds more tools for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, and characteristics of properties (e.g. symmetry), and enumerated classes. SPARQL is a query protocol and language for semantic web data sources.

In this paradigm, documents are "marked up" with semantic information, an extension of the HTML <meta> tags used in today's Web pages to supply information for Web search engines using web crawlers. This could be machine-understandable information about the human-understandable content of the document (such as the creator, title, description, etc., of the document) or it could be purely metadata representing a set of facts (such as resources and services elsewhere in the site).
Note that anything that can be identified with a Uniform Resource Identifier (URI) can be described, so the semantic web can reason about animals, people, places, ideas, etc. Common metadata vocabularies (ontologies) and maps between vocabularies that allow document creators to know how to mark up their documents so that tools can use the information in the supplied metadata.

One of the key advances required to improve the metadata as it currently exists is the need to reduce the number of rules required to describe constrains and assumptions. In addition to the semantic information about the relationship, such as multiplicity of members, constraints on type instantiations, and consequences of participation, is the need to specify which classes always worked together collaboratively, as found in the patterns of interplay or sequence diagrams.

The notion of frameworks was introduced previously, and an important extension of that concept should be explored in the next phase. The idea of associative frameworks (Whitehurst 1997) provides a method to do this. One of the key concepts in addition to grouping together classes that collaborate is the key notion of roles. A framework is a system of cooperating classes that captures the core or essential structure. Constitute a reusable design for a particular class of software applications (Johnson 1988). It standardizes the approach to factoring the functionality of a domain into object classes, and defines the relationships between those classes. Since objects in a framework are collaborating, they are necessarily related by the roles they play in the collaboration process. Frameworks, therefore, are composed of (possibly abstract) classes and inter-class relationships, and include one or more instances of design patterns (associations).

## 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 both within and across architecture communities. 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 in this area and result in measurable progress toward fundamental "better, faster cheaper" LVC environment development goals.

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 would like to 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 Interoperaibilty 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.