Next-Generation Live Virtual Constructive Architecture Framework (LVCAF)

Warren Bizub
Joint Forces Command J7
Suffolk, VA
Warren.bizub@jfcom.mil

Jeffrey Wallace
Carpe Occasio Technology Systems
La Jolla, CA
jwallace@cots-llc.com

Dr. Andy Ceranowicz
Alion Science and Technology
Alexandria, VA
aceranowicz@alionscience.com

Dr. Edward Powell
SAIC
Washington DC
ed@tech.org

ABSTRACT

The challenges faced by today's warfighter – more agile and innovative adversaries in widely varying operational environments – are steadily increasing. To respond effectively, the training and education community must provide timely, relevant and efficient training for operators in a resource constrained environment. The ability to rapidly compose Live, Virtual, and Constructive (LVC) environments is central in addressing this challenge. Current LVC infrastructures lack native support for a variety of critical functions, such as fault tolerance, composition, mediation, load balancing, and information assurance. As a result, several problems are evident: development of custom solutions is pervasive, making them difficult to reintegrate; lack of support for agile re-use (e.g., algorithms or libraries); current Modeling & Simulation (M&S) architectures are not adapted to support net-centric environments or enterprise Service-Oriented Architectures (SOAs); and the mixed architecture LVC environments lead to throughput limitations, and increased latency,that affect scalability.

A new, converged, composable simulation architecture is needed to facilitate re-use of LVC assets, and optimize performance and scalability across the wide range of M&S activities. In order to enable the transformation and evolution, a new approach for harnessing the creative talents of developers supporting the M&S community is required. This paper describes a high-level design, business model, and standards approach for a futuristic LVC Architecture Framework (LVCAF). A mixed business model, consisting of Commercial-Off-The-Shelf (COTS) and open source, is of particular interest. New standards governing such a system, and the Communities of Interest (COIs), needed to implement the approach are discussed.

ABOUT THE AUTHORS

Warren Bizub is the Program 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. After serving in the USAF he moved through various positions in the DoD training community for twenty years as a Project Engineer, Program Manager, In-Service Engineering Supervisor, and Software Support Facility Manager. Before reporting to the USJFCOM, Mr. Bizub was the Science Advisor to the Commander, U. S. Naval Air Forces. He has a B.S. degree in Ocean Engineering, is a graduate of the Navy's Senior Executive Management Development Program, and a MIT fellow of Seminar XXI: Foreign Politics, International Affairs, and the National Interest.

Jeffrey Wallace is the Chief Technology Officer of Carpe Occasio Technology Systems. He has an M.S. and B.S. in Mathematics, University of Illinois Urbana-Champaign, and 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. Mr. Wallace has written over 40 technical papers, and edited several books. He served as the general, or program, chair for numerous International conferences in artificial intelligence, M&S and high performance computing; was the VP for Membership for the Society for Modeling and Simulation International, Board of Directors, and chaired the 50th Anniversary conference. He formed the Modeling and Simulation Professional Certification Commission, and led the team that created the

Certified M&S Professional examination. He also chaired the Emerging Concepts and Innovative Technologies subcommittee for the 2008 Interservice/Industry Training, Simulation and Education Conference (the largest DoD affiliated conference).

Dr. Andy Ceranowicz is a Chief Scientist at Alion and the 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

Dr. Ed 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. He moved to SAIC in 1994, and 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), the Joint Simulation System (JSIMS). He then worked in the intelligence community for two years on architectures for integrating large-scale diverse ISR systems. He has been the lead architect for TENA for eight years now, and is currently working on expanding the applicability of TENA, and integrating multiple interoperability architecture approaches using ontology-based systems.

Next-Generation Live Virtual Constructive Architecture Framework (LVCAF)

Warren Bizub
Joint Forces Command J7
Suffolk, VA
Warren.bizub@jfcom.mil

Jeffrey Wallace
Carpe Occasio Technology Systems
La Jolla, CA
jwallace@cots-llc.com

Andy Ceranowicz
Alion Science and Technology
Alexandria, VA
aceranowicz@alionscience.com

Dr. Edward Powell
SAIC
Washington DC
ed@tech.org

INTRODUCTION

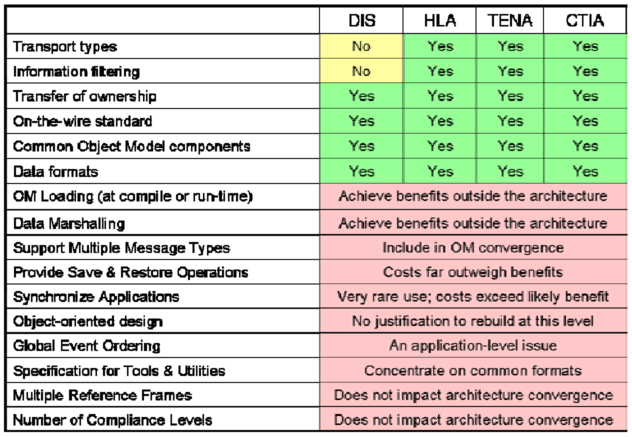
The training community has made significant progress, enabling users to build critical resources through distributed architectures. Some experts characterize the advance as one of the major success stories of the last two decades. Building on early successes, such as SIMNET, different user communities have, over time, evolved infrastructure and protocols tailored for the unique requirements of their community. For example, the Aggregate Level Simulation Protocol (ALSP), dating back to the early 1990's, built on the SIMNET concept of distributed training. ALSP focused on faster-than-real-time simulations, and aggregate level representations, to provide a theater-level experience for battle staff training. Roughly in tandem, SIMNET evolved into the Distributed Interactive Simulation (DIS) standard, IEEE 1278, to provide technical interoperability for entity level simulations.

With the goal of integrating the capabilities of DIS and ALSP into a single architecture, the High Level Architecture (HLA) (Kuhl 2000) was developed. HLA had a broadened focus, beyond the training community, including many new user and exercise requirements. Specifically, DIS and ALSP emerged from the requirements of the training community and the HLA designers recognized the acquisition and analysis communities also had unique requirements for combining hardware, models, and simulations. As such, HLA was the first distributed interoperability paradigm designed from the ground up to support the requirements of the three different communities. However, broad design - to meet diverse requirements - typically sacrifices performance.

The real-time test range communities - who often require precise timing, need high fidelity models, and are extremely sensitive to latencies - experienced performance issues with HLA. Consequently, the Test and Training Enabling Architecture (TENA) was developed as a high-performance, real-time, low-latency interoperability infrastructure. TENA was designed largely to integrate live assets at test range events. Assets included a variety of systems, and numerous models and simulations, capable of generating detailed performance and event data. Similarly, the Army's Common Training Instrumentation Architecture (CTIA) was developed to link assets on an Army training range. CTIA was envisioned to integrate a large number of assets sending narrowly bounded data sets over low bandwidth links to support After Action Review (AAR).

The above architectures are not inherently technically interoperable, although capable of meeting requirements for which they were designed. One approach to achieving the needed architectural interoperability involves converting assets from one paradigm to another. However, this approach can be costly, requires different workforce capabilities, and poses risks. Technical interoperability can also be achieved through other methods, including gateways, translators, bridges, or embedded middleware solutions. However, challenges arise with these methods as well. Problems include violation of latency thresholds, significantly increased complexity, incorrect translation of data, and differences in protocols leading to a lack of robustness.

The DoD Modeling and Simulation Steering Committee sponsored a study called the Live Virtual and Constructive Architecture Roadmap (LVCAR). The study identified a core set of technical recommendations to potentially mitigate the risk in diverging architectures. Table 1 summarizes several important features and capabilities to be considered from a technical perspective that are prime candidates for convergence. The subsequent discussion outlines an approach to consider in solving the problems identified in LVCAR.



	DIS	HLA	TENA	CTIA
Transport types	No	Yes	Yes	Yes
Information filtering	No	Yes	Yes	Yes
Transfer of ownership	Yes	Yes	Yes	Yes
On-the-wire standard	Yes	Yes	Yes	Yes
Common Object Model components	Yes	Yes	Yes	Yes
Data formats	Yes	Yes	Yes	Yes
OM Loading (at compile or run-time)	Achieve benefits outside the architecture			
Data Marshalling	Achieve benefits outside the architecture			
Support Multiple Message Types	Include in OM convergence			
Provide Save & Restore Operations	Costs far outweigh benefits			
Synchronize Applications	Very rare use; costs exceed likely benefit			
Object-oriented design	No justification to rebuild at this level			
Global Event Ordering	An application-level issue			
Specification for Tools & Utilities	Concentrate on common formats			
Multiple Reference Frames	Does not impact architecture convergence			
Number of Compliance Levels	Does not impact architecture convergence			

Live Virtual Constructive Architecture Roadmap

Table 1. LVCAR Technical Convergence Targets

The study explored how cross architecture compatibility could be realized, and forward momentum restored, in moving to the next generation of M&S technologies. The study recommended against another infusion of capital to establish a new architecture, stating it was not warranted and would produce yet another integrating architecture requiring bridging. Rather, a gradual effort to push the existing architectures together was recommended. The study also noted the need to consider industry's potential contribution in moving simulation technology to bridge the gap between the existing architecture environments. A significant question was posed: What characteristics are needed to enable greater productivity and reuse?

The paper begins by discussing lessons-learned and obstacles to reuse. In general, the goal is to provide a deeper understanding of the current challenges. Next, proposed solutions are outlined, composed of two major concepts – an LVC Architecture Framework, and a development environment with capabilities and characteristics suited towards the rapid assembly of LVC exercises and events.

LEARNING FROM PAST LESSONS

The first characteristic that emerges when contrasting the development and expansion of the Web and object-oriented computer languages to that of HLA is the grass-roots origins of the former. The HLA was a top down driven enterprise with limited bottom up support. Without foundational support, many initiatives lose traction and falter. Hence, any new interoperable infrastructure must be accepted by the action officers and upper management through a process of incremental socialization and feedback.

Some industry experts believe the focus on reuse and simulation-linkage has been counterproductive. Interoperability architectures add considerable complexity to simulation (and in general LVC systems) development and operation. Ultimately, the goal is to allow developers to build simulations with less effort and to create more useful products. While reuse provides leverage for easier simulation development, it must be considered a supporting mechanism rather than the primary goal. Making new simulation development easier is the main objective. DoD applications constantly require new and better models for an increasingly wider range of phenomena (e.g., non-kinetic and social effects).

OBSTACLES TO REUSE

Reuse can pose several difficulties. The first obstacle involves locating the models to use. M&S resource catalogs exist, but generally contain large, complicated, and multipurpose simulations. Determining whether the simulations meet the developers' goals is difficult and labor intensive because the available metadata rarely provides all the pertinent/relevant information. The next problem involves obtaining the software. Does it require a memorandum of agreement (MOA), a license, or a fee? Does it require supporting software with an MOA, a license, or a fee? Is a trial version available? This complicated process limits the number of alternatives to be evaluated, let alone reused.

Gaining sufficient understanding of the simulations is necessary before a final selection is made, including:

1. How to operate them
2. Verifying that they work
3. Validating that the models represents the phenomena of interest or can be adapted to do so
4. Understanding how to interface to them
5. Verifying that implementations are compatible with other LVC assets of interest
6. Determining if resource requirements fit the budget.

While advocates for reuse extol the benefits of not having to write new code, they rarely consider the cost of trying to understand the capabilities and operation of other people's software. Advocates normally recommend spending additional funds to produce code and documentation designed for reuse. Time and budget constraints often make this impossible. While most simulation software is reused for many applications, it is limited to the original development team, and those to whom knowledge of the software can be transferred, via apprenticeship and oral tradition. Ironically, additional documentation quickly becomes a maintenance liability. Out of date documentation leads to a frustrating experience for developers.

Once LVC components or simulations are selected, considerable effort remains to interface, adapt, and optimize the resulting simulation to support the target exercise or study. Because the cost of integrating external simulations is generally high, organizations tend to stick with the simulations and federations their developers are familiar with, and upgrade them to include the new phenomena of interest. Integrating external simulations usually requires hiring the external simulation's developers. Organizations that use large federations often have to hire different developers for each of the simulations in their federations, which causes another reuse problem. When the federation needs to represent new systems, doctrine, or areas of operation, the changes often cut across the simulations, and each federate developer has to be paid to make the change. Maintenance for federations is proportional to the number of federates. Furthermore, each simulation typically has different operator interfaces, recovery procedures, and input data. To execute a common scenario, the scenario data and simulation inputs need to be translated into formats understood by each component simulation. In other words, a federation is as difficult to operate as it is to build.

Another problem in federating LVC components and simulations involves semantic incompatibility. Models are abstract representations of the real world. There are many ways abstractions can be developed, resulting in many potential incompatible models. For example, one can build a model (in simulation A) of traffic based on fluid flow; and another developer can model traffic (in simulation B), based on entity interactions. Linking A and B is not easy. Many models do not fit together well, and federating them leads to semantic conflict and simulation anomalies. Largely, the federation process consists of reducing the worst incompatibilities to an acceptable level. When putting a number of simulation components together, we expect the whole to give us more capabilities than the individual parts. Semantic conflict prevents us from realizing the full benefit of joining simulations. Some features are not usable, and restrictions are required for valid results.

Existing integrating architectures support connecting simulations but not reusable modules. Some integrating architectures have different features that can be turned on or off for different federations, limiting interoperability within a single architecture. Due to complexity on most projects, model developers do not deal directly with the integrating architectures. Instead, a federation developer hides it under an abstraction layer so other developers will not have to deal with it. Hence, interoperability architectures require specialized knowledge which hinders reuse.

FIXING THE PROBLEM

The LVC architecture framework (LVCAF) is intended to provide an environment in which the existing LVC architectures can function, providing a forum for convergence and a unified approach to multi-architecture exercises and events. The goals are three fold: 1) to promote convergence, 2) to improve the ability of the community to contribute to the evolution of an enterprise infrastructure and common net-centric data strategy, and 3) to support business model approaches that reward innovation. Key drivers of the LVCAF include non-intrusive value- add for enterprise capability, to simplify application or component integration, and facilitate data integration.

The essential motivation is to improve the economics: optimize the use of human capital; address the scarcity of talent, funding, and time; and be able to quantify return on investment. The current state of affairs limits productivity due to the complications caused by competing architectures, meta-data formats, gateways, bridges, and infrastructures for the multitude of systems. Understanding complexity measurement and management is needed; and coordination between many organizations and cultures is necessary. A unique challenge exists in understanding and managing the human aspect of interoperability and integration. A wide variety of factors must be considered to effect convergence, as is illustrated in Figure 1 below.

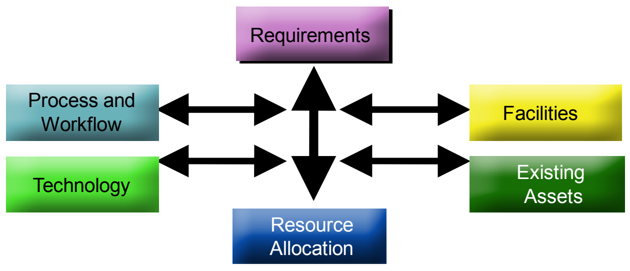


Figure 1. Enterprise Architecture Features

An important feature is the Technology element of the Enterprise Architecture. The subsequent sections provide an overview of technology that can overcome the challenges outlined previously.

LVC ARCHITECTURE FRAMEWORK

The next generation LVCAF will need to include four distinct categories of software, three of which are shown below in Figure 2:

- Tools
- Runtime Framework
- Unique Applications

The Tool applications simplify planning, management, control, and administration of federation execution. The Runtime Framework enables participants in a federation to integrate federation components consisting of LVC assets. The unique applications represent the LVC components to be integrated - to interoperate in a federation. The fourth category is a development environment, described later in this paper as the Integrated LVC Development Environment (IVDE).

The Runtime Framework consists of three main categories, the Core API, and External Module API, and Internal Module API. The Core API consists of interfaces addressing the needs of tools that are used to plan, manage, and control federation components and hardware resources executing in the federation. The Internal Module API has interfaces permitting tight coupling of federation components from a performance and semantic interoperability perspective. Finally, the External Module API is the interfaces allowing loosely coupled federation components that can be rapidly and economically configured.

Internal Federation Management

Ontology Markup and Composition provide a mechanism for describing federation resources and operation to permit automation of composition, execution and control. The incorporation of Information Assurance alleviates the need for developers to implement their own approaches producing patchwork solutions. The Persistence API's and Data Loading modules provide support for event/exercise persistence across multiple processes and support data storage and reload automation. Analytic simulation has not been served effectively by previous simulation interoperability infrastructures.

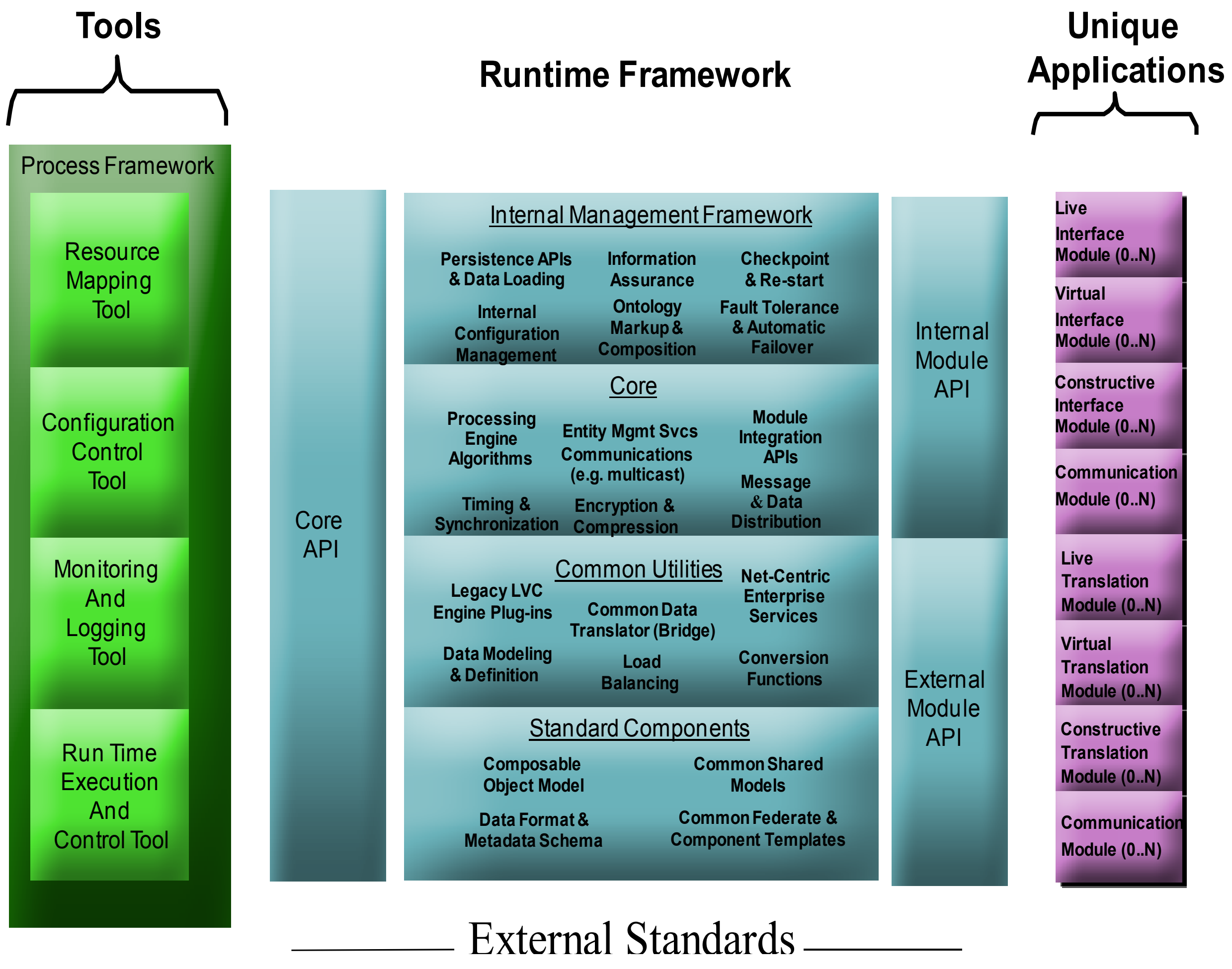

Tools
Runtime Framework
Unique Applications
Process Framework
Resource Mapping Tool
Configuration Control Tool
Monitoring And Logging Tool
Run Time Execution And Control Tool
Core API
Internal Management Framework
Persistence APIs & Data Loading
Information Assurance
Checkpoint & Re-start
Internal Configuration Management
Ontology Markup & Composition
Fault Tolerance & Automatic Failover
Core
Processing Engine Algorithms
Entity Mgmt Svcs Communications (e.g. multicast)
Module Integration APIs
Message & Data Distribution
Timing & Synchronization
Encryption & Compression
Common Utilities
Legacy LVC Engine Plug-ins
Common Data Translator (Bridge)
Net-Centric Enterprise Services
Data Modeling & Definition
Load Balancing
Conversion Functions
Standard Components
Composable Object Model
Common Shared Models
Data Format & Metadata Schema
Common Federate & Component Templates
Internal Module API
External Module API
Live Interface Module (0..N)
Virtual Interface Module (0..N)
Constructive Interface Module (0..N)
Communication Module (0..N)
Live Translation Module (0..N)
Virtual Translation Module (0..N)
Constructive Translation Module (0..N)
Communication Module (0..N)
External Standards

Figure 2. LVC Architecture Framework

Analytic studies tend to use multiple simulations sequentially rather vs in parallel. For example, a model may examine a situation in high resolution and produce data to drive models to operate in lower resolution with wider scope. Current interoperability architectures retain the training world's focus on presenting a single fused experience. All the models run in parallel and exchange data while running. To be applicable to many analysis problems, the LVCAF should address data exchange and the coordination between sequential simulation executions. Internal Configuration Management provides a method for querying and managing the configuration of components required to instantiate a federation

The Fault Tolerance and Automatic Failover functions support the execution of a federation. The module constructs federations to automatically address component failure, allowing the federation to run without interruption. Checkpoint capabilities permit simulation state saving transparently, during run time. Restart, via, the defined checkpoints, can return to the simulation (in case of unrecoverable events).

LVCAF Core,

Entity Management Services permit various entities, represented by the federation components, to share states with the whole system. The Processing Engine Algorithms control the entire LVCAF instantiation. The algorithms and implementations should be optimized for various hardware and system resources. Timing and Synchronization are modular, permitting the LVCAF core to be configured with several timing, synchronization algorithms, and implementations, for user optimization.

The LVCAF Communications framework permits the LVCAF middleware to utilize multiple communication protocols simultaneously (e.g., shared memory utilized between federation components executing on a multi-core computer, TCP/IP between networked computers, and several communication protocols with operational or live participants such as Link-16 and Joint Tactical Radio System (JTRS). Message and Data Distribution supports various methods to optimize communications for particular hardware configurations. The methods adapt to many network loading and traffic scenarios.

The Module Integration API's provide easy integration of new components to the system. They simplify timing, synchronization, and message passing between federation components and resources. Encryption and Compression provide third party security, and ensures security of messages and data to optimize available band-width via advanced compression techniques.

Common Utilities

Data Modeling and Definition simplify the process, and shorten integrate time with all components in the federation, into an interoperable system, including LVC component modules. Similarly, a Net-Centric Enterprise Services – The ability to utilize and comply with DISA's Net-Centric Enterprise Services (NCES), Net-Enable Command Capability (NECC), and Net-Centric Data Strategy (NCDS) should be built into the LVCAF middleware.

The Legacy LVC Engine and Processing Plug-Ins is a significant aspect. In order to ensure a smooth transition from legacy LVC federation components, a set of plug-ins can be built to permit a high performance interface between the new LVCAF middleware and legacy components. Interfaces to DIS, HLA, TENA, and CTIA systems will be available. Load Balancing functions provide the federation to dynamically adjust allocation of certain federation components to hardware resources, based on overall computational and communication loading.

Unique Applications

The Live Interface Modules enable live system resources to be integrated into a federation, with special emphasis on timing and synchronization, data and message transfer - especially analog-to-digital and digital-to-analog conversions. Virtual Interface Modules provide an ability to integrate next-generation realistic virtual components into federations facilitated by the internal module API's. Constructive Interface Modules improve integration of large scale portrayals of many phenomena. Scalability and fault tolerance/automatic failover are prime considerations.

In certain scenarios, the internal communication modules provide live system communication protocols integration, particularly for embedded training and experimentation, in which the boundary between LVC components is obscured. Live translation modules allow rapid integration of loosely coupled live components into a federation. The external module API's represent a set of simplified interfaces for situations in which limited timing, synchronization, message, and data distribution are adequate. Hence, the approach permits rapid integration. Virtual translation modules ease the integration of virtual systems not requiring tight coupling and a moderate volume of data exchange. Constructive translation modules permit loosely coupled constructive federation components with a simplified set of interfaces for timing, synchronization, message, and data distribution. The external communication modules are provided to quickly integrate live system communication protocols, in which integration with operational Warfighters is required.

Tools

Resource mapping tools assess federation execution requirements against the hardware assets available. The resource mapping tools can manage and optimize network bandwidth and processor loading. Configuration control tools provide a system-wide capability to assess and manage all aspects of federation components, and hardware resources, available to execute the federation.

Monitoring and Logging – The ability to monitor all aspects of federation components and the hardware resources executing the federation are needed. In addition to monitoring, all the information should be logged in a way that permits ready analysis and inspection via a suite of query utilities. A single central logger is a scalability bottleneck. Hence, distributed logging and queries are essential for large scale simulations. The user should also have a choice between the efficiency of binary logging and the understandability of text logs. Support for calculating metrics and visualization of results at runtime and for post processing is needed.

Finally, the Run-time execution and control tools manage and control all federation components and hardware resources executing the federation. GUIs tools are needed to provide interfaces for exercise participants.

SOLVING THE KNOWLEDGE PROBLEM

The LVCAF provides all the pieces and tools required to connect components from a large number of sources into a single combined simulation federation. Yet the previous analysis of obstacles suggests - even if a painless way existed to connect any set of simulations, the primary problem would remain: acquiring sufficient knowledge of the available simulations to select, adapt, and operate them. An approach to the problem has been historically employed in other domains, but not the M&S community. This is due to the unique problem in M&S presented by the need to represent time and system evolution. The approach has been behind subroutines, databases, and object-oriented methodologies, which is to decompose the problem up into small components which can then be more easily understood and reused.

Currently, the only reusable assets in the M&S community are complete models and simulations typically composed of thousands of components. Ideally, even if developers followed code reuse development practices, a significant problem still exists in determining how all the pieces fit together and interact. Many modern software techniques, such as event-based and thread-based programming, make the software difficult to figure out compared to the traditional static calling sequences of procedural programming. Moving toward loosely coupled M&S, and more generally LVC, assets with smaller meaningful components is positive.

A fine-grained, component-based development approach would enable more developers to contribute innovations. Intuitively, the more LVC components become available, the more useful combinations can be integrated to more easily solve problems. The infrastructure must encourage LVC developers to create smaller LVC components that can integrate together, and then make those pieces available to others. However, in order to be able to quickly find and arrange very large numbers of LVC components into useful systems, the reality that human intellect alone is insufficient must be faced. Composition assistance using machine intelligence will be required. Unfortunately, the coding of M&S components is generally obscure, and does not permit what is being modeled to be understood. The domain concepts are represented in the software in a way that the relationship to the real-world phenomena is obfuscated by the machinery required to make the simulation work. The domain is barely recognizable to humans much less to machines. In the ideal case models would be documented along with the real-world referents in an implementation independent format.

The larger software development community has realized this and moved to the Model Driven Architecture (Gasevic 2006) to capture how software components interact with each other and with the world in formats such as the Unified Modeling Language (UML) and other related techniques such as the Schlaer-Mellor method dating back to the 1990's (Schlaer 1991, Wallace 1998, Mellor 2002). In the C4I community the DoD Architecture Framework (DODAF) is now required for systems documentation. These techniques have a more understandable structure, but are still human-oriented representations. Representations manipulated and matched in intelligent ways by computers are needed.

Dating back to the 1990's (Berners-Lee 1999), numerous R&D efforts have converged to create a body of knowledge and technology now known as the Semantic Web (Allemang 2008). The goal of these efforts was to describe and represent the meaning, or semantics, embedded in web pages in a way that software agents can understand and interact with them. This is very similar to the capability that is required to understand M&S and LVC components, and provides a hierarchy of tools capable of capturing semantics of M&S assets and how they represent and relate to the world. These tools include:

- RDF, the resource description framework
- RDFS, the RDF schema language
- SPARQL, a query language for RDF
- OWL, the web ontology language
- SWRL, the Semantic Web rule language

RDF provides a way to build distributed database tables defining how concepts relate. RDFS permit machines to infer relations not specified explicitly. SPARQL allows distributed databases to be queried to retrieve explicit and inferred data. OWL provides additional inference mechanisms; and SWRL allows the application of automated logic to the semantic representations and the database of knowledge.

Any knowledge representation presents a grounding problem (e.g., the meanings of the terminal nodes need to be defined). As such, the words suggest real world meanings to humans, but software does not have human experience to call upon. To software, an 'F-16' is not the same as an 'F16'. Fortunately, the Semantic Web addresses exactly the same problem. The Semantic Web community is solving the problem by building publicly available ontologies describing the real world, including these equivalency and translation elements. This permits machine inference to match and interface, if we can agree on common ontologies to use, or specify the mappings between the ontologies used by different models. Employing different ontologies to provide a mapping between equivalent concepts improves developer efficiency. This allows innovation to continue simultaneously with standardization.

The Semantic Web developers are creating tools needed to describe model components for computer reasoning. Thus, model compatibility, translations between data exchange elements, and developer conflicts can be identified. However, building ontologies is more difficult than writing natural language documentation. If time does not permit general documentation, building ontologies will pose a problem. Part of the answer can be found in the data exchange specifications of current interoperable simulations. Interoperability architectures like HLA and TENA require machine readable specifications of the data exchange elements or 'object model' for the federation. These can be converted into model specific ontologies and related to standardized ontologies for the real world, enabling evaluation on how two simulations can interoperate with each other.

A primary goal of the objective interoperability infrastructure is to encourage model developers to break up simulations and LVC assets, and create smaller model components. The result will be that most of the meaning of the composite model will be captured in the interfaces between the model components, and making the additional step to documenting the internals of the models manageable. The availability of public and authoritative ontologies for the subject matter domains will cause developers to use them in the process of new model and LVC assets creation, since less work is required via resuse. The same ontologies can also be applied to sharing source data and scenarios between simulations. The ontologies can eventually be used to support human instruction, decision-making, and real systems engineering. Enormous potential for reuse exists, if common ontologies are employed.

DEVELOPMENT ENVIRONMENT REQUIREMENTS

The interoperability architecture needs to gain sufficient adoption to generate a network effect for LVC and simulation components creation and reuse. It should be attractive for the majority of LVC system and simulation users, including those who do not need to network assets together. The interoperability infrastructure must replace general purpose programming languages, and specialized simulation languages, as the medium for expressing models. There is no reduction of effort for the developer creating a single use standalone simulation, if it is necessary to first express the model in a general purpose programming language and then interface it to an interoperability framework to make it work. Furthermore, the infrastructure has to capture the meaning of the models in such a way that machines can intelligently make use of it for matching and linking with other models without further burdening the developer.

This leads to the requirement to build an Integrated LVC Development Environment (IVDE) that can capture the user's model specifications at a high level and implement them as executable LVC systems or

distributed LVC system components that combine new and existing model components together with the LVCAF to create new products. In reality, this will encourage innovation, while allowing specialized solutions where sufficient benefit exists. What is needed is a way to define families of integrated development environments that share ontologies and can build compatible models.
Despite the previous focus on the standalone LVC system or simulation user, to achieve the stated goals the IVDE must be capable of producing products that are executable in a wide range of environments. The resulting LVC components need to be easily changed to work in a networked environment based on protocols compatible with existing HLA, TENA, DIS and CTIA environments. The LVC components also need to be usable as parts of embedded training systems, decision support systems and command and control systems. Capturing the model and component definitions in an architecture independent form provides the best opportunity to compile the model definition into executables compatible with different systems.

This in itself is a huge reuse advantage over trying to combine modules already specialized for a particular environment. The challenge is that the code generation software becomes more complex with increasing levels of abstraction from the executable software. Accordingly, there is an optimal level of model specification that will depend on the code generation capabilities available. Success depends on not being too far away from that optimal level.

Most modern computer languages have benefited enormously from having the ability to download libraries of source code that can be reused. The source code is publicly available on the web (known as open source code). For example, the R language (Chambers 2008) enables the developer to just enter a command and the interpreter will go out find a mirror, fetch the software, and install it onto the developer's computer. This ease of reuse is needed in the objective environment.

The R language is especially interesting from the perspective of modeling and simulation. It is a statistical data processing language that allows users to not only access and apply statistical models; it also provides access to a large number of source data sets. This is relevant to modeling and simulation because right now most LVC system and simulation developers have very little access to source data which could provide the necessary context for understanding some models, as well as support better verification and validation. Our IVDE should give the developer similar access to a wide variety of models, data, and solution tools without the impediments of manual searching of registries and repositories, requesting software, getting approvals, and waiting for it to arrive. The ontologies provide the key for finding what the user needs. A necessary component of the IVDE is a semantic search engine for relevant ontologies, models, simulation components, LVC components, and support modules. This semantic search engine must have the

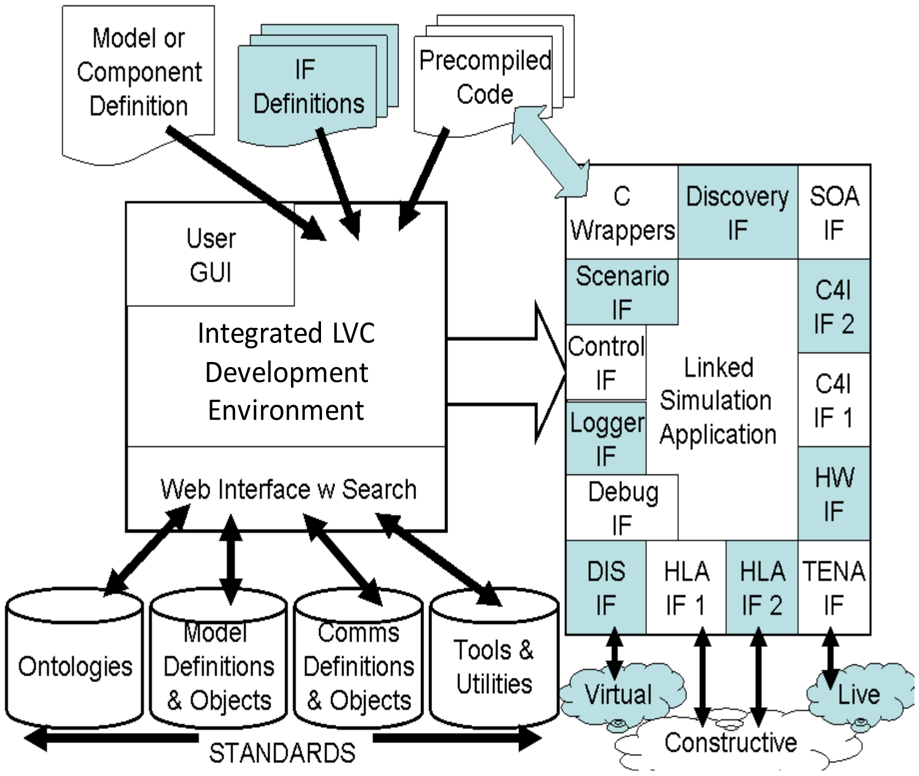


Figure 3. Integrated Simulation Development Environment

ability to express what the system must do, what assets are available to use, and the ability to reason about these inputs and produce solution suggestions in the form of suggested components and data sets.

INTEGRATED SIMULATION DEVELOPMENT ENVIRONMENT

The next generation of LVC interoperability tools is envisioned to be accessed via IVDEs (there may be many such systems) as shown in Figure 3. An IVDE will first capture the developer's descriptions of the desired model, simulation, or LVC component. The IVDE will then accept specifications of the interfaces for the systems or federations to be employed, plus the interface specifications of any foreign code modules that the developer wishes to include, and use this information to produce an LVC or simulation application.

In the process of creating the application the IVDE will help the developer discover and reuse:

- Ontologies
- Simulation Models
- LVCAR components
- Custom communications interfaces
- Custom tools and utilities

These resources will be located on a network, either the World Wide Web, or perhaps a DoD or organization intranet, for sensitive systems. The LVCAF components do not need to be located in one place or delivered on a DVD. The IVDE can be implemented as a service oriented web application. This makes the capture of simulation artifacts much easier than relying on users to upload their products. Using the mechanism of Semantic Web ontologies, the IVDE will search the appropriate network to find and download the resources that fit the specifications of the application.

The ontologies will also help the developer reuse standardized concepts that already exist for the domain of reference. If the developer finds the available conceptual models to be inadequate, the IVDE allows the developer to define new ontologies and map them to the existing ones. The key improvement is that the developer does so knowing what is already out there. The ontologies are then used to create translation routines between all the subcomponents in that application and to any external LVC components, applications, or federations with which the new application must interact.

The developer can also download custom utilities and tools for runtime federation control, monitoring and logging, resource mapping and configuration control. The IVDE will produce linked applications containing only those models, tools, and components needed for a given purpose and optimized for a particular environment. Federations can be built by having participants compile with the same federation specification. Running in new environments would require recompiling with a specification for that environment.

IVDEs should make it easier to integrate models at a fine level of granularity to help avoid many semantic conflicts by linking only models from independent domains. Most semantic conflict occurs because simulations are complete so that they can execute in a standalone mode. Thus they each need to have their own representation of all relevant domains, such as terrain. Currently, if two simulations are linked their representations of terrain will probably be different, inducing anomalies and what are known as "fair fight" issues. However, if models from different domains are employed, those problems do not usually occur. For example, if a model of shopping is created along with a model of traffic, and then combine them to produce a model of a shopping mall, there should be few incompatibilities. In general, linking existing simulations, and other LVC assets, almost always results in overlapping domains. Clearly, the focus should be on assembling and linking models from different domains into new simulations and LVC components. This will lead to far less semantic conflict than the current practice of linking entire LVC and simulation applications, and therefore better results.

Of course, just building an IVDE is not sufficient. There needs to be enough content to attract users. This means that the vast store of M&S material currently existing cannot be abandoned. At a minimum, users will want to take advantage of existing simulation resources. That means that the IVDE will need to be able to link in existing LVC assets and simulation code. This will require the users to specify the interface to those code modules in an ontology, similar to wrapping functions in one language so that they can be called from another, but with a higher level of meaning. The same capability would be used to interface to specialized simulation hardware such as might be required in virtual simulations. Variants of the IVDE are envisioned to support construct of virtual simulations as well as hardware in the loop systems – that is serving the entire LVC community.

INFRASTRUCTURE OBSTACLES

Besides obstacles to reuse that have to be overcome, the obstacles to fielding and getting acceptance of a new interoperability infrastructure or IVDE must be considered. When people propose a Swiss Army Knife approach to a problem, there is natural skepticism as to the achievability of the approach, its ease of use, its reliability, and its efficiency. These are valid questions for the IVDE. Achieving the IVDE will require research and development in a number of related areas, including the integration of ontology-based reasoning engines with code-generation capabilities.

The progress being made in Semantic Web technologies indicates integration of an IVDE is possible in the near future. For ease of use, the IVDE should allow users to employ the system at a variety of levels of difficulty with proportional levels of capability. At the entry level users might access existing LVC assets, simulations, and data, then customize and run them. At the next level they might create standalone LVC applications or simulations. The key is that knowledge to execute each task needs to be carefully separated in such a way that as users enter more complex environments they have the mechanisms to get tasks done.

Reliability requires a consistent underlying structure that is properly matched to the target functionalities of the system. To achieve this, ontologies can be leveraged to create an underlying model of the architecture which can support semi-automated testing. It is also essential that to provide sufficient debugging tools for the users' applications. It will not be practical for the user to pull out a C++ debugger and try to figure out where the design failed. The price of creating high-level definition languages is the need to also provide all the tools required to debug the output, including logging, assertions, watch points, and break points.

Many complicated interface solutions sacrifice efficiency and scalability to generality. Almost anything can be interfaced as long as the scope of the simulation is small and only a few runs are required. Moving the definition of the models up to as high a level as practical allows the minimization of what is included in the resulting executables. For example, if interfacing with DIS systems is not required, there won't be DIS components in the generated product. Compiling the model specifically for particular architectures with significantly more knowledge about the model to be simulated than possessed by conventional compilers should also enable significant optimizations.

Availability is a critical issue. If the LVC component or simulation developer has to pay a substantial amount of money to obtain the infrastructure, the number of users will be limited, as will the network effect. Ideally the infrastructure would be used as part of most college courses that include simulation development. As such, a free infrastructure would be best. In the same spirit, the ability to leverage all the talent in the community is also essential – another aspect of the network effect. It is essential that mechanisms are found to engage academia, industry, and government by providing ways that each can contribute to the solution in a manner compatible with their operating principles. This leads to the realization that open source development must be considered as a viable approach.

STANDARDS SOLUTIONS

Clearly the ontologies used for automating the search and integration of components for LVC systems and simulations must be standardized. The ontologies will make it possible to get away from the developer intensive task of finding and understanding the available models. Machine support of this process is essential and standard ontologies are needed to make it work. Standards for the interfaces and functions for the tools and utilities in the LVCAF are also critical. Without standardization, dealing with different offerings from many sources would become intractable. Standards for user interfaces would allow the creation of uniform control interfaces while utilizing the products of multiple developers. This would allow LVC systems and simulations to be run by fewer operators.

Other areas for potential standardization include making sure that the outputs from the code generation processes are linkable between different IVDEs. We need to standardize the format and required content of the specifications used for interfaces so multiple IVDEs can employ the same specifications. Similar standardization is required for the way scenarios and parametric data are specified. In general, creating environments where competing products can be easily interchanged is desirable. In particular, enabling commercial ventures to create plug-ins and add-ons to the base system should be a goal. In order to achieve this end state, a progression thorough multiple standards as the community learns how to solve problems until solutions are created that are good enough to justify a single approach will likely be the case.

BUSINESS MODEL SOLUTIONS

A complicated technology thrust like this, which seeks to change the landscape of modeling and simulation and the LVC community is clearly beyond a single project or organization. This will be realized through the contributions of many different organizations over an extended period. The business model needs to support this and avoid shutting out any group. The core of the environment, enough to create useful LVC components and simulations that can be used in production environments should be free and open source. This opens up the technology to universities for research purposes and for use in courses that train future practitioners in the art of simulation. It also brings in open source developers and innovation. While the core should be open source, it should not prevent government and commercial organizations from building on the core and creating value-added restricted-distribution products from it. Government organizations may want to protect certain innovations, such as information assurance features, from foreign/hostile agents. The key is for the open source core to contain enough functionality to remain viable as a standalone application.

In general, a large body of existing software is excluded when open source is required. For some components a new Government Open Source license might be created. The majority of government software development is conducted under contracts that simply provide for free use for government purposes. This approach appears to lead to stovepipes where development resides with the primary contractor and does not encourage reuse and research investment across contractors. For each application a secondary development contractor needs a government sponsor to certify the application often ruling out Internal Research & Development investment.

In most cases, new product development is not built on GOTS applications because secondary companies are not guaranteed the right to reuse the software in new projects. Nor do they know if their changes will be incorporated into the government baseline. One thing to explore is a possibility to provide the contractor a license that allowed them to maintain their own version of the Government software and propose it for any new Government contract they bid on, and then perhaps more reuse of Government software would likely occur. How to integrated new software developed products (as open source) by the contractor back into the baseline application requires future investigation as well.

Technology transfer concerns, especially with developed software related to defense applications, can be extremely complicated and are beyond the scope of this paper. However, when dealing with DoD applications we need to recognize that moving toward an open source environment will require close scrutiny to intellectual property and technology transfer issues.

COMMUNITIES OF INTEREST

The expectation exists that user and developer organizations will form communities of interest around different applications and aspects of the IVDE. From the application side, a major problem exists that Semantic Web technology can’t solve: figuring out the relative value of different models and components. The anticipated solution is that different LVC components and simulation domains will develop communities of interest (COI) that will evaluate and rank components that are applicable to their interests. Another expectation is that COIs devoted to different LVC asset and simulation user interfaces would emerge.

For example, one user interface might specialize in expressing models as Petri nets and be graphically oriented while another might be procedure and text oriented. They could both produce compatible models but their users would want to form different communities of interest to share techniques. Virtual simulation users, hardware in the loop users, and live system and simulation users will also want to share their experiences in their own COIs. The underlying LVCAF technologies will also require a number of development COIs. There will probably be one or more compiler groups, a group for the semantics of models, and perhaps others on the semantics of scenarios and source data. In addition, the development and use of IVDEs will produce more than one COI.

SUMMARY

DoD needs faster and more efficient methods for producing new simulation environments to support its operations. Interoperability approaches have diverged and multiplied. A common LVC Architecture Framework can provide an environment in which the existing LVC architectures can function, a forum for convergence and a unified approach to multi-architecture exercises and events. Higher level integration support is required to allow simulation developers to take advantage of all interoperability approaches under a family of compatible Integrated LVC Development Environments. By focusing these IVDEs on making simulations easier for all simulation developers, we can achieve the network effect that will motivate widespread adoption. Semantic Web tools will allow us to catalog, find, and interface simulation components with much more automation than is currently available, lowering the time and cost of fielding new LVC environments.

ACKNOWLEDGEMENTS AND DISCLAIMER

The authors would like to thank Barbara and Jaclyn Hannibal for their editing, graphic arts, and production support of this paper.

The opinions expressed in this article are those of the authors and do not represent the opinions or views of the Department of Defense or the United States Government.

REFERENCES

Allemang, D. and Hendler J. (2008) *Semantic Web for the Working Ontologist, Effective Modeling in RDFS and OWL*, Boston: Morgan Kaufmann

Berners-Lee, Tim, and Fischetti, Mark (1999) *Weaving the Web,* Harper Collins Publishers.

Chambers, J. M. (2008). *Software for Data Analysis,* New York: Springer.

Gasevic, D., Djuric D., and Devedzic, V. (2006). *Model Driven Architecture and Ontology Development*, Berlin: Springer.

Kuhl, F., Weatherly, R., and Dahmann, J. (2000). *Creating Computer Simulation Systems, An Introduction to the High Level Architecture*, Upper Saddle River: Prentice Hall PTR.

Mellor, Stephen, and Mark Balcer (2002) *Executable UML, A Foundation for Model Driven Architecture*, Addison Wesley.

Shlaer, Sally, and Stephen Mellor (1991) *Object Lifecycles: Modeling the World in States*, Yourdon Press.

Wallace, Jeffrey; G. Leonard; L. Peterson; A. Vagus; C. Kropp, 1998. "Using IMPORT to Develop Wargames," In The *Proceedings of the 1998 Object-Oriented Simulation Conference*, The Society for Computer Simulation, San Diego, CA, January 11-14, 1998.