Utilization of Service Oriented Architecture (SOA)-based Commercial Standards to Address Live, Virtual, Constructive (LVC) Interoperability Challenges

Dr. Gary Allen
Joint Training Integration & Evaluation Center
Orlando, FL
gary.allen@us.army.mil

Lucas Schroeder
The MITRE Corporation
Orlando, FL
lschroeder@mitre.org

ABSTRACT

The concept of using Service Oriented Architecture (SOA)-based software technologies is not new and is being eyed with keen interest by many in the simulation industry. However, no extant program of record can afford to put their program at risk on an unproven approach, no matter how promising. The 2008 DOD study "Live Virtual Constructive Architecture Roadmap (LVCAR) Final Report recommends to, "Take actions that can reduce or eliminate the barriers to interoperability across the architectures" (Henninger 2008). As an early step towards addressing the LVCAR recommendation, a 'SOA Outlook' pilot effort was developed to determine if commercial SOA architectures, software and principles are an appropriate solution space for achieving LVC interoperability."

This paper will describe the pilot effort designed around the use of open standards wherever possible and attempts to illustrate SOA principles like composition and re-use. A common data abstraction layer in the application server provides an abstraction of the storage mechanism through the Java Persistence API (JPA) standard and allows for non-system-specific storage of shared data. Integration with existing legacy systems uses a two-part adaptor / plug-in architecture where the adaptor connects directly to the existing infrastructure and communicates with its plug-in counterpart inside the application server infrastructure. The pilot also includes a sample of other services that would be required for a complete interoperability framework. The SOA pilot successfully provides a limited interoperability framework based on the constraints of the use case selected and the level of effort involved. While not a 'silver bullet', this successful pilot demonstrates the applicability and viability of SOA-based architectures inside the LVC interoperability domain.

ABOUT THE AUTHORS

Dr. Gary Allen has worked in various aspects of modeling and simulation for the past 30 years. He was a member of the team that founded the Training Simulation Center for I Corps at Ft Lewis Washington (1980), Director of the Simulation Training Branch at the US Army Intelligence Center and School, Ft Huachuca, AZ (1989-1992), and Project Director for the TACSIM Intelligence Simulation and part of the design group that initiated the Aggregate Level Simulation Protocol (ALSP). From 1996 – 2008 he was the US Army Liaison Officer to the German Military Research and Development Agency in Koblenz, Germany. Currently he is Project Manager for the DoD High Level Task, 'Live, Virtual, Constructive Architecture Roadmap Implementation' project. 1978 MS - Telecommunications Systems Management, School of Engineering, University of Colorado (Boulder) 1989 PhD – Instructional Technology, University of Kansas (Lawrence) Dr. Allen is a member of the Phi Kappa Phi National Honor Society, 1999 graduate of the Army War College, and an Acquisition Corps Level III Certified PM

Lucas Schroeder is a Sr. Simulation and Modeling Engineer with the MITRE Corporation. Prior to joining MITRE Corporation, he has worked as a Software Developer for various companies in the defense sector working on serious games and asset management systems. His area of expertise lies in system of systems data alignment and interoperability. Lucas has a BS and MS in Computer Engineering (Focus area - Intelligent Systems) from the University of Central Florida.

Utilization of Service Oriented Architecture (SOA)-based Commercial Standards to Address Live, Virtual, Constructive (LVC) Interoperability Challenges

Dr. Gary Allen
Joint Training Integration & Evaluation Center
Orlando, FL
gary.allen@us.army.mil

Lucas Schroeder
The MITRE Corporation
Orlando, FL
lschroeder@mitre.org

INTRODUCTION

Current simulation event engineers have a range of architectural capabilities open to them. They can select a "minimalist" intercommunication architecture, providing little more than a communications service, or they can utilize a more complex architecture featuring multiple advanced services such as time and data management. They can also choose to rely on multiple architectures, as occasionally necessitated by the mix of simulation systems that will be combined in the event.

Mixing architectures, however, is not easily achieved: bridges must be installed, gateways developed, and data exchange models (i.e., object models) rationalized and composed. All of this effort is "over and above" that necessary to join the simulations systems that use a common architecture and is frequently viewed as a baseless requirement, none of which would be necessary if multiple architectures were not involved. As a result, the OSD M&S Steering Committee (MSSC) commissioned a study to examine various aspects of M&S development and make recommendations that could improve architecture interoperability.

The Live, Virtual, Constructive Architecture Roadmap (LVCAR) study began in April, 2007. The MSSC recognized that M&S capability had greatly advanced, routinely enabling the linkage of critical resources through distributed architectures. In part, the success was predicted on an iterative and evolutionary development of the intercommunication architectures, including progressive capabilities enhancements supporting more varied application of the technologies across expanding user domains. While the architectures displayed impressive capability to meet needs as designed, they were not implemented with a focus on ensuring architectural compatibility. Thus, each requirement to connect systems using different architectures within a single simulation event was accompanied by substantial design and engineering effort to achieve cross-architecture interoperability. Given this environment, the LVCAR study was chartered to:

> "…*methodically and objectively develop a recommended roadmap (way forward) regarding LVC interoperability across three broad areas of concern: notional definition of the desired future architecture standard, the desired business model(s), and the manner in which standards should be evolved and compliance evaluated.*" (Henninger 2008)

Based on these characterizations of the problem, the study recommendations emphasized a two-front philosophy. First, near-term actions were necessary to ease the problem of architecture integration. Integration should be made transparent, so that users would interact with a seamless "architecture of architectures." Second, a longer-term goal emphasized an evolutionary process of CTIA, HLA, and TENA architectural interoperability.

LVCAR-IMPLEMENTATION PROJECT OVERVIEW

The LVCAR-Implementation (LVCAR-I) project's aim is to explore organizational and structural (e.g. use of standards) options to better: 1) manage LVC architecture interoperability; 2) create reference models to focus data and service reuse efforts; 3) reduce LVC architecture divergence and tool proliferation; and 4) explore emerging technology issues related to future LVC architecture performance and requirements. The planning, development, and execution of LVC events are universally recognized to be expensive by any measure. Also, the M&S community lacks the agility to support unforeseen events without great difficulty. Given this situation, the objective of LVCAR-I is to reduce overhead and thus improve the ability to

construct and conduct timely LVC events. Described another way, the goal for LVCAR-I is to get M&S support inside the military operations decision cycle.

The project leads have taken a holistic approach to organization and definition of an acquisition strategy. Fundamentally, LVCAR-I is designed to work in an environment where there are many different factors and incentives that influence decisions, including willingness to change and the adoption of technical solutions. Understanding these factors and their effects are as important to the success of the project as the technology advances themselves.

Even though the main emphasis of the LVCAR-I project is on gaining efficiencies that support interoperability without creating new architectures the original LVCAR Study did take a cursory look at the potential of Services Oriented Architectures (SOA) and the potential for using that technical paradigm to not only provide a way to connect various federates but also provide connectivity to the Command and Control (C2) systems that are commonly used in M&S events (Henninger 2008) The reference to SOA in the LVCAR study provided the justification to apply resources in creating the LVCAR-I SOA Pilot project described in this paper.

The LVCAR-I SOA Pilot development work was an effort undertaken to prove out the architecture concept of using SOA as a method to achieve LVC interoperability with a non-trivial dataset as an 'architecture of architectures'.

SOA Pilot Objectives

At a high level the objective of the LVCAR-I SOA pilot was:

> *Demonstrate the ability to interoperate disparate LVC systems by developing a pilot infrastructure that uses current SOA best practices.*

This main objective was supported by two secondary objectives:

1. Implement open standards for interoperability.
2. Document lessons learned for the larger modeling and simulation community.

Architecture

In order to support the objectives of the pilot development effort an architecture concept was developed, which would become the basis for the actual prototype implementation. This conceptual architecture is depicted in Figure 1.

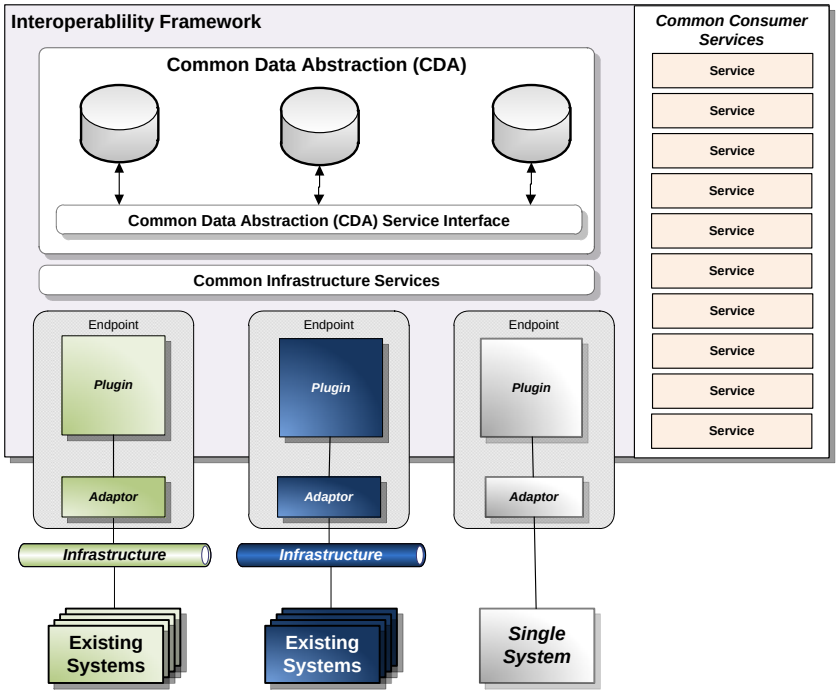

Figure 1. Conceptual Architecture

MAJOR ARCHITECTURAL COMPONENTS

The conceptual architecture can be described as being made up of four major components: Common consumer services, Common Data Abstraction, Common Infrastructure Services, and Endpoints.

Common Consumer Services

Common consumer services are services designed to offer capability out to end users, or consumers of data. Examples of common consumer services may be data feeds providing a Situational Awareness picture of the LVC operating environment, or a centralized technical monitoring service detailing operational status.

Traditionally these capabilities have been provided independently of central organization and implemented in each component of the larger enterprise LVC solution. These common consumer services provide a number of benefits over the traditional methodology of system independent implementation. The most important benefit is that these services help provide for a single, cohesive view into the combined operating environment provided by all the connected systems. In addition, the use of common consumer services also lowers the development and maintenance costs associated with such functional capabilities while lowering the entry requirements for new systems to integrate into the larger LVC enterprise, since much of the development has already been done.

Common Data Abstraction

The Common Data Abstraction (CDA) provides the heart of the interoperability framework. It provides a system agnostic central data model and the run-time population of this central data model is persisted into a database using Object Relational Mapping (ORM) capabilities. The ORM service interfaces are the only piece of the data model that is visible and accessible to outside users, thus the interface layer shields end users from the details of the central data model and will provide a number of benefits described in following sections.

Common Infrastructure Services

This collection of services is differentiated from common consumer services in that the infrastructure services provide capabilities that are more often than not utilized by other services. Examples of common infrastructure services include such capabilities as Extensible Style Language Transformation (XSLT) services, data filtering services, and Java Messaging Service (JMS) messaging services.

Endpoints

Endpoints are a collection of two subcomponents, a plug-in that resides within the Interoperability Architecture, and an adaptor which resides outside the architecture and connects to legacy infrastructures. These two components are specific to each legacy system, or system of systems that wishes to be connected through the interoperability architecture. In this manner an endpoint provides much of the same functionality that is currently found in gateways and bridges that are being used to solve interoperability requirements today. However the development of an end-point is faster than the development of a point to point bridge or gateway in that the common services provide large chunks of functionality which are able to be utilized ‘out of the box’.

PROTOTYPE DEVELOPMENT

In order to prove out the concepts presented in the Conceptual Architecture, a single use case was selected for implementation in the prototyping effort (see Figure 2). The use case selected was the development of a prototype interoperability layer to connect the Army Joint Land Component Constructive Training Capability (JLCCTC) Multi-Resolution Federation (MRF) and JLCCTC Entity Resolution Federation (ERF) training systems together and facilitate the sharing of data across the prototype framework. The MRF and ERF federations were chosen due to familiarity, accessibility of required resources, and the collection of technical challenges presented in achieving interoperability between these two systems.

As an example, even though both systems are essentially High Level Architecture (HLA) federations, they use very different object models. ERF uses an entity-centric data model based loosely on the Distributed Interactive Simulation (DIS) standard, while MRF utilizes an aggregate unit-centric data model based upon the older Aggregate Level Simulation Protocol (ALSP). The MRF federation is a time-managed federation in which all simulations coordinate the advancement of simulation time and simulation time may differ from wall clock time, while ERF on the other hand runs in real-time. These major differences along with a host of other smaller challenges indicated that the use case of providing interoperability between these two families of systems would provide for a valid test environment.

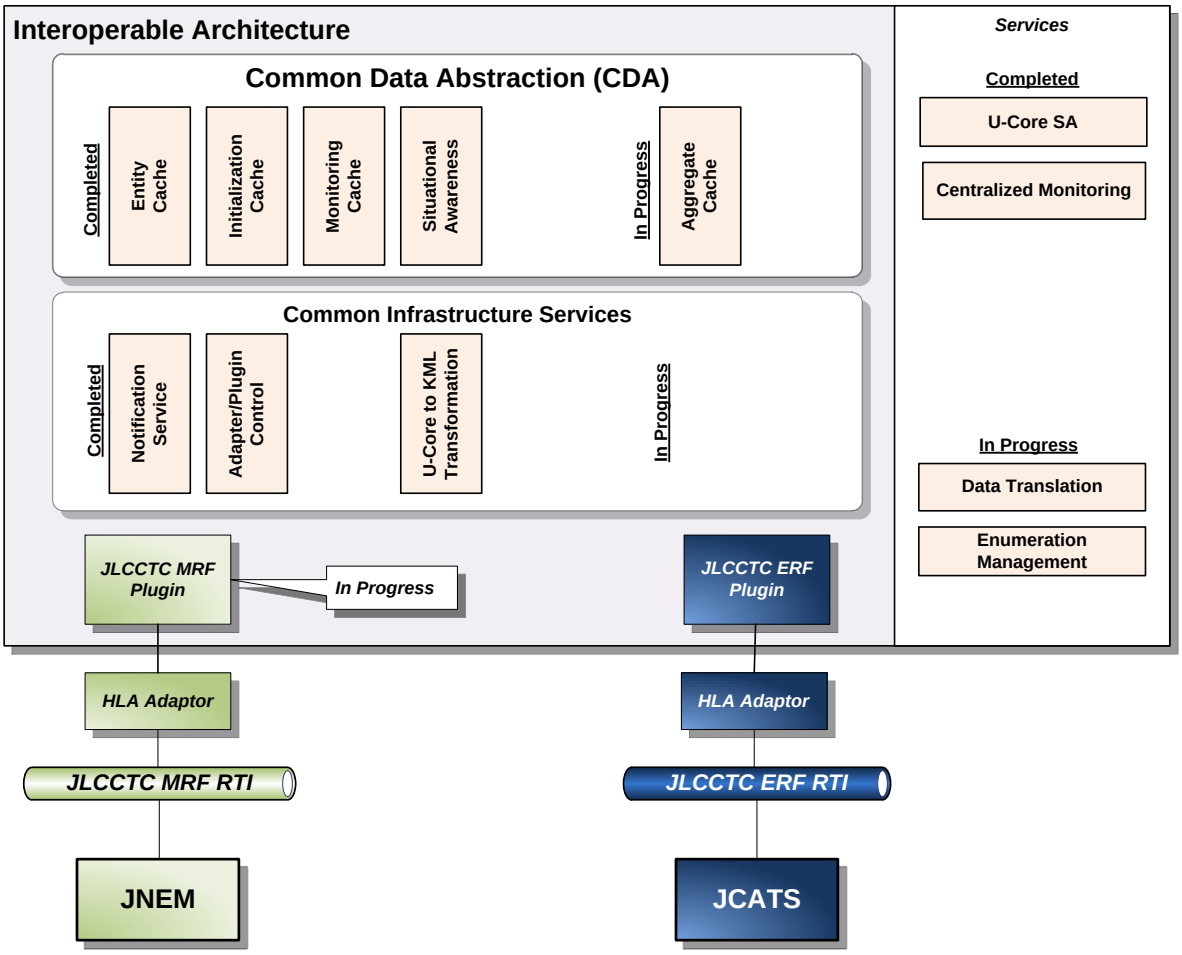

Figure 2. Prototype Implementation

This prototype implemented a number of services in order to demonstrate SOA concepts as well as illustrate potential solution paths to some of today's larger interoperability challenges. In some cases optimal design patterns were not implemented in order to highlight SOA concepts and how they might be implemented inside this architecture to provide benefit over today's traditional interoperability designs.

Common Consumer Services

For the prototype development two common consumer services were developed, along with two more services that were in progress at the end of the development effort. The first service developed provides a status of the various connected endpoints, to be used for a centralized tech control monitoring capability. This is a capability that is often missing in current interoperability enterprises.

The second service developed was a Situational Awareness service which provides ground truth for all entities within a geographic bounding box in a UCore XML format (UCore 2010). This situational awareness feed can be utilized to populate a Common Operating Picture, and as a demonstration of capability this situational awareness feed was displayed inside the Google Earth environment for visualization purposes.

This kind of capability can be used for a variety of applications including feeding a Common Operational Picture (COP) to Command and Control (C2) devices, providing an overview of the operating environment for exercise controllers and planners, and providing a methodology for outside commercial viewers to be used to visualize the operating environment (e.g. Google Earth, Marble, or NASA WorldWind).

In addition a substantial amount of work was done on the development of an enumerations translation service. This particular service provides a set of common methods for data producers and consumers to be able to go to a centrally managed service to determine how to translate from native representations to common data abstraction representations. It will provide a single common solution to a problem which all gateways and bridges face, and under current practices are usually implemented individually and on a point-to-point basis.

While the service was completely developed and unit tested, full integration of the use of the service into the existing endpoints has not been completed at this time. The current implementation of the enumeration service and its planned usage by the plugin components of endpoints can be seen in Figure 3. However this expected use does not prohibit other systems (i.e. individual simulations that are members of a federation connected to the interoperability architecture) to utilize this capability to further enhance the data translations occurring even inside constituent systems of systems.

Common Data Abstraction

The common data abstraction developed for the pilot effort encompasses the minimal set of data required to share object state across the framework for all entities in the operating environment. The common data abstraction consists of a number of service interfaces which exist outside the physical data model. Object representations in these service layers are persisted into a physical storage medium via the commercial Java Persistence API (JPA), which is a standard defined for Object Relational Mapping (ORM). In some cases primitive services, like those that enable read and write of entity state data are utilized by other services providing reading of Situational Awareness data, and Read / Write of Aggregate object state data.

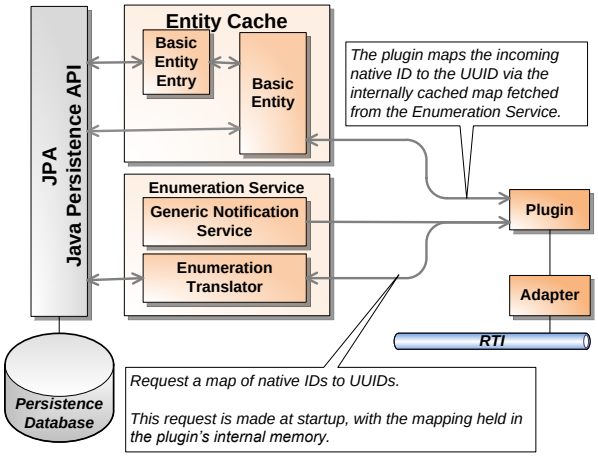

Figure 3. Enumeration Common Consumer Service

The common data abstraction implemented in the prototype effort also looked into partitioning of data for performance reasons. In the prototype, the service collection for dynamic entity state date is placed in one Enterprise Java Bean (EJB) containing attributes such as Health State, Location, Speed, Heading, etc… A second EJB was created to hold more of the static (or less dynamic) data such as C2 Role, Unit Name, Parent Unit, etc… This particular partitioning methodology seems to help with performance characteristics and should be investigated with more quantitative testing.

Common Infrastructure Services

For the prototyping effort – only two common infrastructure services were required. The first is a generic notification service which allows other services to post updates or service messages to specified JMS topics. This allows for a standard inter-service communication protocol, and can be re-used from implementation to implementation.

The second being an XSL Transformation (XSLT) service which converts a UCore XML data stream into a Keyhole Markup Language (KML) data stream and is used to transform the Situational Awareness data feed into a format which commercial tools (such as Google Earth) can ingest and visualize.

Endpoints

Two endpoints were developed as part of the prototyping effort, an HLA endpoint for the JLCCTC ERF Federation, and a second HLA endpoint for the JLCCTC MRF Federation. Each of these endpoints follows the same design paradigm, illustrated in Figure 4. While each endpoint as a whole has to be built specifically for the system which connects to it, components within the endpoint are reusable. For example, in the pilot development it was possible to reuse the Adaptor portion of the endpoint (with a different configuration file) for both endpoints since the adaptor is the component that does the protocol translation part of the effort and both constituent systems are HLA federations.

The Plugin components designed as part of the endpoint are responsible for providing a rules engine which specifies how to translate incoming and outgoing data feeds from native representations of the connected system into the system agnostic representation stored in the CDA and vice versa. In the current prototype implementation the rules engine is hard coded to provide the required functionality, but it is desired to replace this implementation with a standard business rule management system in order to facilitate evolution through Subject Matter Expert (SME) input that is not wholly dependent on developers for implementation.

ARCHITECTURE BENEFITS

Modularity / Agility

One of the big advantages of implementing a SOA based architecture is the advantage provided by modular development (Green et al. 2008). This modular nature in turns gives the enterprise a much needed agility when it comes to future requirements, reconfigurations, and evolution. This modularity and agility really support the way many existing systems are currently doing program development.

This modularity helps solve a large interoperability problem that is often seen in current interoperability practices. When trying to coordinate a large enterprise system with many constituent systems, all of which have their own development cycles and development timelines, it becomes hard to coordinate development of new capabilities and schedule integration testing when each program is working on their individual parts of a collective capability on their own timelines.

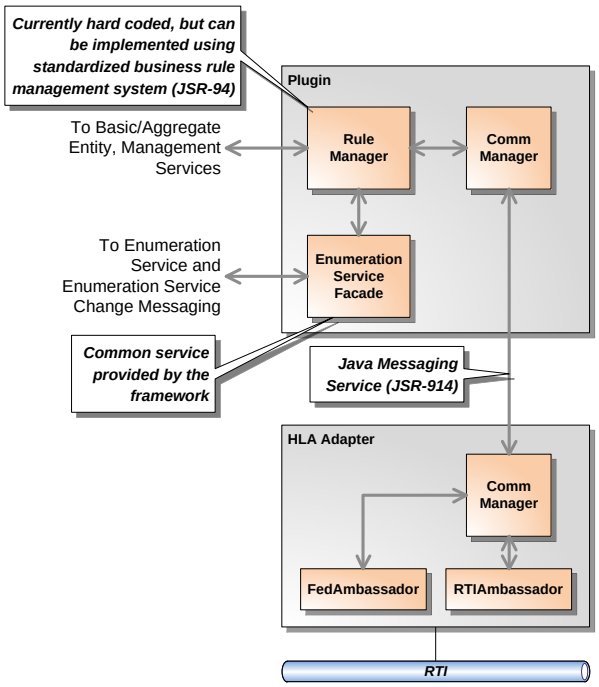

Figure 4. Endpoint Design

These issues of integration scheduling become most noticeable in current operations during the evolution of the central data model, since changes to the central data model require all constituent systems in the enterprise to step up to the new model at the same time (with some minor exceptions). However due to the development of the CDA as a service interface layer over the physical/logical data model, this issue is partially alleviated by allowing multiple versions of service interfaces to exist simultaneously. See Figure 5 for an example evolution and how the CDA service layer provides excellent support for design evolution.

Reusability

Reusability often comes hand in hand with the idea of a modular architecture, but is often difficult to achieve in a meaningful manner. In a SOA-based architecture reusability can be achieved in a number of different fashions. In the first case reuse is demonstrated in the prototype through the developing of a reusable component, the HLA adaptor that was utilized in both endpoints.

Another successful demonstration of reusability is achieved in a different manner by using the SOA concept of composition of services. For example in the prototype development effort, the Entity State services of the CDA are utilized by the Situational Awareness Service, the Aggregate Unit Service, and individual consumers such as the plugin for the ERF federation. This type of reuse dictates that a single service can provide its capabilities over and over again to achieve different functions for its end consumers.

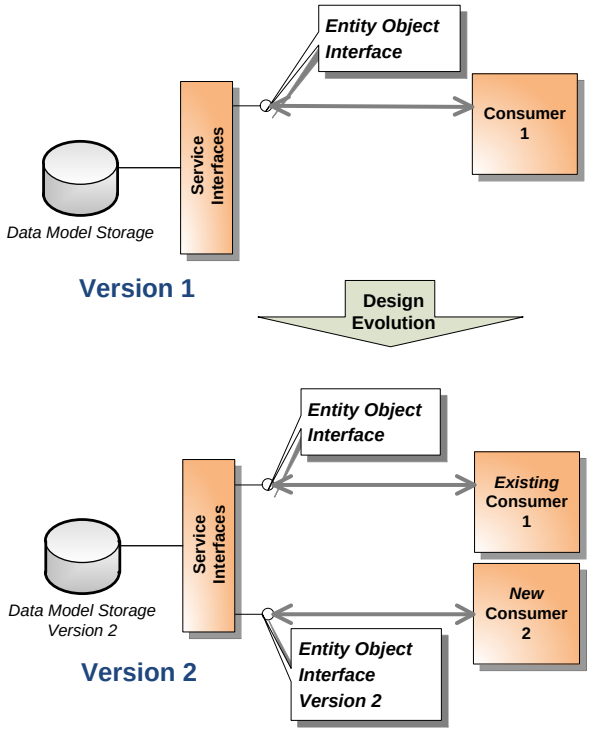

Figure 5. Example Design Evolution

Common Implementation

The idea of common implementation strategies in the SOA Pilot architecture provides for a number of benefits:

1. All end data consumers receive the same source data, leading to a more consistent view of the operating environment across the constituent systems
2. Lower development cost when both systems engineering effort and development effort is paid for once (in the implementation of a common service) rather than being done by each constituent system separately
3. Integration issues are easier to identify when all constituent systems are using the same common service, rather than their own custom implementation

One last benefit of a common implementation strategy is the ability to gain new capabilities that were not available under an architecture where similar capabilities are spread across multiple constituent systems. For example, in the enumeration service capability the concept of editing or creating mappings while the system is running can be accomplished. This provides that all consumers of the data will automatically be updated through the notification system of the changes without having to take any action on their part, thereby allowing for centralized management of these enumerations mappings and runtime updates. This is not something that is currently allowed in LVC interoperation events due to each constituent system using its own implementation strategy for building mappings into its own bridge or gateway.

Commercial Software and Standards

Most of the benefits already discussed can be achieved through clever systems engineering and architecture designs utilizing current interoperability paradigms. However, the SOA Pilot architecture's use of commercial software and standards provides for a number of other benefits which cannot easily be achieved using Government Off the Shelf (GOTS) developed solutions.

The first major benefit of using commercial software solutions in the SOA Pilot architecture is the availability for scaling, high performance, high availability options for operation. This allows the architecture to scale from single box implementations for small scale exercises, or one-off experiments to server clusters supporting large scale exercises in a distributed environment. These scaling capabilities come out of the box and may require some engineering design into the enterprise network layout and routing but are essentially free capabilities available for utilization depending on the use case requirements.

A second benefit of the use of commercial software is the fact that it is in wide usage, so it has been hardened, battle-tested, and is constantly being provided with bug fixes and security patches, without maintenance or development expense to the LVC community. In addition this large user base provides for a wealth of developer knowledge available to LVC developers in building their own implementations.

CONCLUSION & OBSERVATIONS

During the course of developing the SOA Pilot prototype implementation, a number of salient observations have been noted and are worth sharing with the community.

The first observation is that the adoption of a SOA-based architecture is not a silver bullet towards achieving interoperability. In order to create interoperability an amount of "core work" still has to be done in order to translate data and adapt messages from protocol to protocol (Brooks 1987). However through the use of good systems engineering the SOA-based architecture does allow for a smaller expenditure of effort in achieving this "core work". For example look at Figure 6 to see how by repositioning where "core work" is being done; pieces of the effort can be reused thus saving on effort expenditures.

Another observation worth noting is that many critics of SOA architectures state that performance is an issue for the scale of large LVC exercises. During the course of the development of the SOA Pilot prototype a standard JLCCTC ERF Validation Scenario, numbering around 108K entities was used for all testing and development. During testing on previous generation standard workstation hardware, the SOA Interoperability Architecture never became a bottleneck, even when running on a flat network. Further performance testing is required to fully determine the performance characteristics of such an architecture, and is currently under investigation.

The last point worth noting during the development of the SOA Pilot prototype was that the proposed SOA LVC Architecture is not an "all-or-nothing" proposition. Existing implementations can gradually

move towards utilizing such an architecture in an evolutionary fashion and will help facilitate adoption by the larger LVC community. One of the underlying constraints on the development of the SOA LVC was that existing LVC solutions would not have to be replaced entirely, but could slowly migrate towards such an architecture. It is recognized that wholesale replacement of existing systems is not a realistic option.

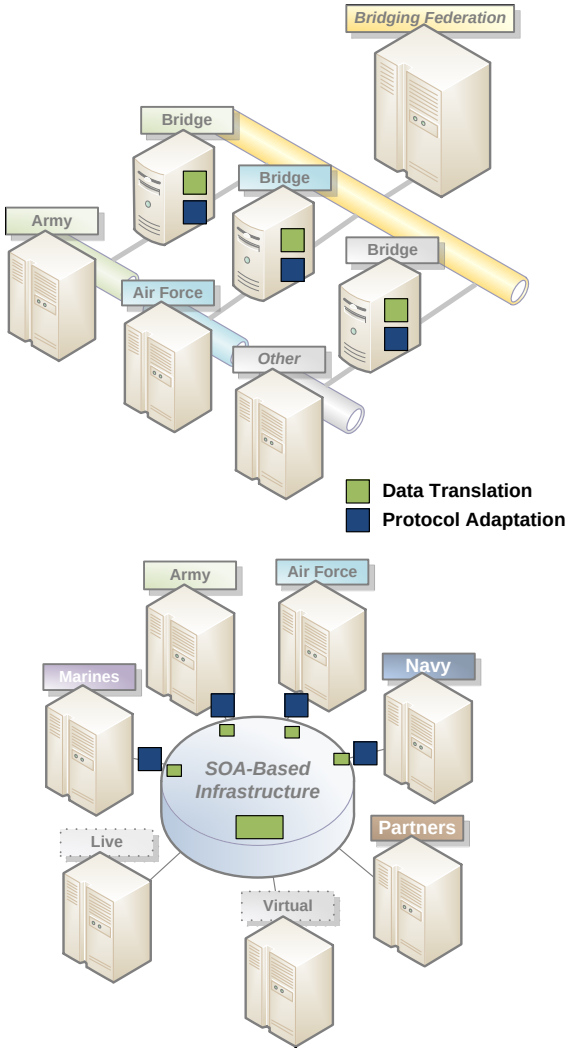

Figure 6. Traditional Interoperability Architecture vs. SOA Pilot Architecture - Core Work Locations

While we consider the SOA Pilot Project to be a significant success there is a question that all M&S event designers must ask: "Is SOA a good fit to meet our goals?" To help provide some insight to that question a team of investigators from the Johns Hopkins University Applied Physics Laboratory (JHU/APL), under the auspices of the LVCAR-I Project, looked at the pros and cons of employing SOA in DOD M&S environments. (Drake 2011) Instead of making an overarching judgment on SOA technology the resulting study provides a benefits/barriers analysis and an accompanying set of criteria that can be applied that will assist M&S event designers in deciding whether SOA technology is an advisable way to meet the event's objectives. (Drake 2011) The benefits and barriers analysis is too extensive to be covered in this paper other than to say that SOA is not a panacea for all distributed, interactive simulation events so users need to be judgmental when looking to apply this architecture.

Of the many things we learned from this project and its related studies there are a number of items that have been covered in technical papers and were reinforced in this project: (Schaffner 2006)

1. SOA is not only a technical approach (governance and business models are key components)
2. Applying SOA is an evolutionary approach (you grow a SOA environment)
3. SOA can be built using the components of an existing architecture
4. Services can be built from legacy systems
5. Success is aided by the use of recognized standards

During the course of the SOA Pilot we made a significant effort to market our activities by hosting a number of 'community meetings' where we invited members from the M&S Community of Interest that include all the services and various agencies. The purpose of the meetings was to inform attendees of our activities, solicit input, and provide the potential for an off ramp of what we accomplished. These events were telecast to help increase participation and we had 20+ attend each of four meetings. The project team has also been involved in sharing lessons learned with technical representatives from the US Army Live, Virtual, Constructive – Interoperability Architecture (LVC-IA) Program of Record, the Joint Forces Command Live, Virtual, Constructive Architecture Framework (LVCAF) project, and members of the Program Executive Office – Simulation, Training, and Instrumentation (PEO-STRI) engineering staff including Dr. James Blake (PEO STRI). In addition, the JHU/APL team has provided lessons learned seminars at various M&S venues and are in the process of developing a lessons learned 'e-learning' Mobile Application. In the end the LVCAR-I SOA Pilot team measure of success will be the continued pursuit of this effort to the ultimate benefit of our warfighters.

REFERENCES

Brooks, Frederick P. (April 1987). "No Silver Bullet: Essence and Accidents of Software Engineering," Computer, Vol. 20, No. 4 pp. 10-19

Drake, D, et al (March 2011). *Service-Oriented Architecture Application to Live-Virtual-Constructive Simulation: Approach, Benefits, and Barriers (NASD-R-2011-025).* Unpublished Report available from the Modeling & Simulation Coordination Office.

Henninger, A, et. al. (September 2008). *Live, Virtual, Constructive Architecture Roadmap (LVCAR) Final Report*. Unpublished Report available from the Modeling & Simulation Coordination Office.

Green, J., Besemer, D., Butterworth, P., Clement, L., Green, J., Ramachandra, H., Schneider, J., Verdervoot, H. (April 2008). *Service Oriented Architecture: Getting it Right*. Sunnyvale, California: Westminster Promotions

Schaffner, B (September, 2006). *10+ Things You Should Know About Service Oriented Architecture (SOA)*. Online Article: TechRepublic http://www.techrepublic.com

UCore (2010) *UCore Main Page* Retrieved on 9-16-2010 from http://ucore.gov

Wikipedia (2010) *Service Oriented Architecture* Retrieved 09-10-2010 from http://en.wikipedia.org/wiki/Service-oriented_architecture