A Systems Engineering Perspective on the Development and Execution of Multi-Architecture LVC Environments

Robert Lutz
Johns Hopkins University
Applied Physics Laboratory
Laurel, MD 20723
robert.lutz@jhuapl.edu

Roy Scrudder
Applied Research Laboratories
The University of Texas at Austin
Austin, TX 78713
scrudder@arlut.utexas.edu

Ronda Syring
Johns Hopkins University
Applied Physics Laboratory
Laurel, MD 20723
ronda.syring@jhuapl.edu

Mikel D. Petty
University of Alabama - Huntsville
Huntsville, AL 35899
pettym@uah.edu

Jake Borah
Aegis Technologies
Orlando, FL 32826
jborah@aegistg.com

John Rutledge
Trideum Corporation
Huntsville, AL 35805
jrutledge@trideum.com

Keywords:
DSEEP, LVC, systems engineering, multi-architecture, overlay

ABSTRACT: *The Live Virtual Constructive Architecture Roadmap (LVCAR) defines a time-sequenced set of actions and activities which collectively improves LVC interoperability and lowers the technical and cost risks associated with the development of multi-architecture simulation environments. One of the key activities identified in the Roadmap is the establishment of a common systems engineering process for multi-architecture LVC applications. The stated goal of this process is to improve communication and collaboration among the disparate architecture communities that must interact and work together toward common goals during a multi-architecture development. During LVCAR Phase II (Roadmap Implementation), this product was developed as an overlay on top of the IEEE P1730 Distributed Simulation Engineering and Execution Process (DSEEP). This paper reports on the mapping of multi-architecture pertinent issues to the various activities defined in the DSEEP, and provides examples of the guidance that is provided for how users of this process may address the various issues. Finally, this paper discusses current plans for standardization of this product under SISO and the IEEE.*

1. INTRODUCTION

Modeling and simulation (M&S) has long been recognized as a critical technology for managing the complexity associated with modern systems. In the defense industry, M&S is a key enabler of many core systems engineering functions. For instance, early in the systems acquisition process, relatively coarse, aggregate-level constructive models are generally used to identify capability gaps, define systems requirements, and examine/compare potential system solutions. As preferred concepts are identified, higher-fidelity models are used to evaluate alternative system designs and to support initial system development activities. As design and development continues, very high-fidelity models are used to support component-level design and development, as well as developmental test. Finally, combinations of virtual and constructive M&S assets are frequently used to support operational test and training requirements. Note that other industries (e.g., entertainment, medical, transportation) also make heavy use of M&S, although in somewhat different ways.

The advent of modern networking technology and the development of supporting protocols and architectures have led to widespread use of *distributed simulation*. The strategy behind distributed simulation is to use networks and support simulation services to link existing M&S assets into a single unified simulation environment. This approach provides several advantages as compared to development and maintenance of large monolithic stand-alone simulation systems. First, it allows each individual simulation application to be co-located with its resident subject matter expertise rather than having to develop and maintain a large stand-alone system in one location. In addition, it facilitates efficient use of past M&S investments, as new, very powerful simulation environments can be quickly configured from existing M&S assets. Finally, it provides flexible mechanisms to integrate hardware and/or live assets into a unified environment for test or training, and it is much more scalable than stand-alone systems.

There are also some disadvantages of distributed simulation. Many of the issues related to distributed simulation are related to *interoperability* concerns.

Interoperability refers to the ability of disparate simulation systems and supporting utilities (e.g., viewers, loggers) to interact at runtime in a coherent fashion. There are many technical issues that affect interoperability, such as consistency of time advancement mechanisms, compatibility of supported services, data format compatibility, and even semantic mismatches for runtime data elements. The capabilities provided by today's distributed simulation architectures are designed to address such issues and allow coordinated runtime interaction among participating simulations. Examples of such architectures include Distributed Interactive Simulation (DIS), the Test and Training Enabling Architecture (TENA), and the High Level Architecture (HLA).

In some situations, sponsor requirements may necessitate the selection of simulations whose external interfaces are aligned with different simulation architectures. This is what is known as a *multi-architecture simulation environment*. There are many examples of such environments within the Department of Defense (DoD) [1]. When more than one simulation architecture must be used in the same environment, interoperability problems are compounded by the architectural differences. For instance, middleware incompatibilities, dissimilar metamodels for data exchange, and differences in the nature of the services that are provided by the architectures must all be reconciled for such environments to operate properly.

Because of perceived increases in the number of multi-architecture simulation events anticipated in the future, along with the associated increase in costs, the DoD sponsored an initiative to examine the differences among the major simulation architectures from technical, business, and standards perspectives and to develop a time-phased set of actions to improve interoperability within multi-architecture simulation environments in the future. This initiative was called the Live-Virtual-Constructive Architecture Roadmap (LVCAR). The first phase of this effort began in the spring of 2007 and continued for approximately 16 months. The result of this activity was a final report and supporting documentation that collectively totaled over 1000 pages [1].

A major conclusion of the LVCAR effort was that migrating to a single distributed simulation architecture was impractical, and thus multi-architecture simulation environments would remain the state of the practice for the foreseeable future. One of the key actions recommended in the LVCAR Phase I Report was the establishment of a common systems engineering process for the development and execution of multi-architecture simulation environments. The widely reported issue in this case was that when user communities of different architectures were brought together to develop a single multi-architecture distributed simulation environment, the differences in the development processes native to each user community were creating a persistent barrier to effective collaboration. That is, since these communities had to work together toward common goals, differences in the practices and procedures these communities typically use to build new simulation environments were leading to misunderstandings, misinterpretations, and general confusion among team members. This was impacting risk from many different perspectives. Thus, as part of the implementation of the Roadmap (LVCAR Phase II), a task was initiated by the Modeling and Simulation Coordination Office (MSCO) to develop this common process view for multi-architecture development and execution. This multi-architecture process was developed by a core systems engineering process team (the authors of this paper) with input from a broad cross-section of simulation subject matter experts.

2. DSEEP OVERVIEW

To develop this common systems engineering process, the core team and subject matter experts reached consensus that leveraging and modifying/extending an existing systems engineering process standard was preferable to building an entirely new process description from scratch. Early in the project, the systems engineering process team considered several generalized and widely recognized systems and software standards (e.g., EIA-632, ISO/IEC 15288). However, the team (with input from the subject matter experts) decided that direct reuse of any process standard outside of the M&S domain would require a significant degree of tailoring, consuming resources that could be better applied in other ways. For that reason, the team selected an emerging Institute of Electrical and Electronics Engineers (IEEE) standard (IEEE 1730) as the foundation for the desired process. The name of this standard is the Distributed Simulation Engineering and Execution Process (DSEEP) [3].

The DSEEP represents a tailoring of best practices in the systems and software engineering communities to the M&S domain. The DSEEP is simulation architecture-neutral, but it does contain annexes that map this architecture-neutral view to DIS, HLA, and TENA terminology. A top-level view of the DSEEP is provided in Figure 1.

In the DSEEP document, each of the seven steps is further decomposed into a set of interrelated lower-level activities. Each activity is characterized according to a set of required activity inputs, one or more output products, and a list of recommended finer-grain tasks.

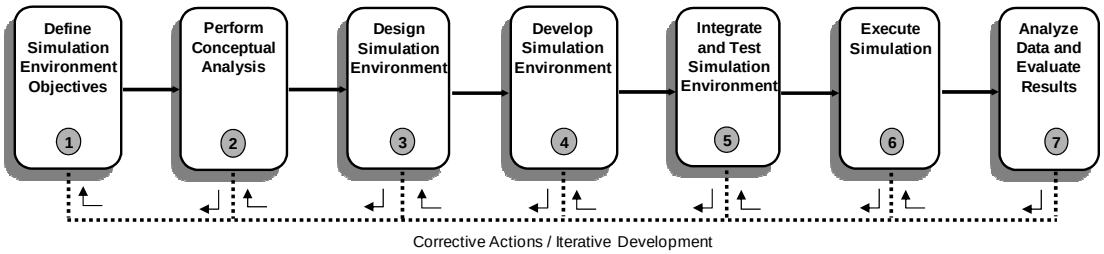

Figure 1. Distributed Simulation Engineering and Execution Process (DSEEP), Top-Level Process Flow View

Although these activity descriptions are identified in a logical sequence, the DSEEP emphasizes that iteration and concurrency are to be expected, not only across activities within a step but across steps as well.

While the DSEEP provides the guidance required to build and execute a distributed simulation environment, the implicit assumption within the DSEEP is that only a single simulation architecture is being used. The only acknowledgement that this assumption may be false is provided in the following paragraph from DSEEP Activity 3.2 (Design Simulation Environment):

"In some large simulation environments, it is sometimes necessary to mix several simulation architectures. This poses special challenges to the simulation environment design, as sophisticated mechanisms are sometimes needed to reconcile disparities in the architecture interfaces. For instance, gateways or bridges to adjudicate between different on-the-wire protocols are generally a required element in the overall design, as well as mechanisms to address differences in simulation data exchange models. Such mechanisms are normally formalized as part of the member application agreements, which are discussed in Step 4."

Clearly, additional guidance is necessary to support the development of multi-architecture simulation environments. However, the major steps and activities defined in the DSEEP are generally applicable to either single- or multi-architecture development. Thus, the DSEEP provides a viable *framework* for the development of the desired process, but it must be augmented with additional tasks as necessary to address the issues that are unique to (or at least exacerbated by) multi-architecture development. Such augmenting documentation is often referred to as an *overlay.* The tasks defined in the desired overlay collectively define a "how to" guide for developing and executing multi-architecture simulation environments, based on perceived best practices for issue resolution. This resulting overlay is the "Guide for Multi-Architecture Live-Virtual-Constructive Environment Engineering and Execution" [2].

3. DSEEP OVERLAY

The overlay was developed starting with an extensive literature search. Over 70 articles and publications were identified that provide case studies of multi-architecture developments, the problems that occurred and issues that were encountered in these developments, and the actions development teams took to address the problems and issues. The majority of the relevant publications were found in the archives of the Simulation Interoperability Workshops and the Interservice/Industry Training, Simulation, and Education Conference, but other conference archives and publications were used.

The result of the literature search was a set of issues for multi-architecture design, development, and execution. In several cases, multiple publications addressed the same problem or issue. In those cases, a common issue description was developed. Next, each issue was mapped to an activity within the DSEEP. Where an issue impacts multiple steps in the DSEEP, the DSEEP step predominantly affected was chosen. Many of the DSEEP activities yielded multiple issues.

For each issue, a recommended action (or set of actions) was identified to ameliorate or resolve the issue. In some cases, these recommendations came from the literature search. In other cases, the core systems engineering team were able identify appropriate actions based on their extensive experience in DIS, TENA, HLA, and the use of these three architectures in multi-architecture-supported events.

The result of this effort was the identification of 41 unique issues with one or more recommended actions to address each issue. The distribution of issues across DSEEP activities is shown in Table 1. Not all DSEEP activities have issues associated with multi-architecture environments. This is especially true for the early and late stages of the DSEEP. This was not an unexpected

Step	(1) Define Simulation Environment Objectives **(2 issues)**	(2) Perform Conceptual Analysis **(2 issues)**	(3) Design Simulation Environment **(22 issues)**	(4) Develop Simulation Environment **(7 issues)**	(5) Integrate and Test Simulation Environment **(6 issues)**	(6) Execute Simulation **(1 issue)**	(7) Analyze Data and Evaluate Results **(1 issue)**
Activities	Identify User/Sponsor Needs **(no issues)** Develop Objectives **(no issues)** Conduct Initial Planning **(2 issues)**	Develop Scenario **(no issues)** Develop Conceptual Model **(no issues)** Develop Simulation Environment Requirements **(2 issues)**	Select Member Applications **(2 issues)** Design Simulation Environment **(14 issues)** Design Member Applications **(1 issue)** Prepare Detailed Plan **(5 issues)**	Develop Simulation Data Exchange Model **(2 issues)** Establish Simulation Environment Agreements **(1 issue)** Implement Member Application Designs **(2 issues)** Implement Simulation Environment Infrastructure **(2 issues)**	Plan Execution **(2 issues)** Integrate Simulation Environment **(no issues)** Test Simulation Environment **(4 issues)**	Execute Simulation Environment **(1 issue)** Prepare Simulation Environment Outputs **(no issues)**	Analyze Data **(no issues)** Evaluate and Feedback Results **(1 issue)**

Table 1. Distribution of Multi-Architecture Issues across DSEEP Activities and Steps

result. The early stages of defining requirements needs, objectives, scenarios, and conceptual models are largely architecture independent. The same is true for the latter stages of executing in the simulation environment and analyzing data. As one would expect, the majority of issues impacting the DSEEP step are associated with designing the simulation environment. The activities associated with developing, integrating, and testing the simulation environment are also areas where a significant issues were identified.

In the DSEEP, a set of inputs, tasks, outcomes are provided with description of each activity. The inputs are information, personnel, or other resources necessary to conduct the activity. The tasks are a set of concise statements describing what has to be done to successfully complete the activity. The outcomes are the resulting information or resources from conducting the tasks. Given that the "Guide for Multi-Architecture Live-Virtual-Constructive Environment Engineering and Execution" is intended to be an overlay to the DSEEP, the same structure was followed. For each DSEEP activity, a set of additional inputs, tasks, and outcomes needed to support multi-architecture environments was developed. The inputs, tasks, and outcomes from the DSEEP were not repeated in the guide. As with the DSEEP, these inputs, tasks, and outcomes are list of one-line descriptions, with few exceptions. The inputs and outcomes were either taken from the literature search results or developed by the core systems engineering team based the information and resources implied by the activity descriptions. The tasks lists in the resulting guide were drawn directly from the activity descriptions.

Not all of the issues identified in the "Guide for Multi-Architecture Live-Virtual-Constructive Environment Engineering and Execution" document tie to all of the architectures. The final step in the creation of the guide was identifying which issues tie to which architectures. This was done through a cross reference table--issues as rows and architectures as column with check marks identifying which architectures are impacted for each issue.

4. OVERLAY EXAMPLES

Given the limited space in a paper of this type, it impossible to provide a complete elaboration of all of the issues identified for multi-architecture environments. Thus an exemplary subset of those issues has been selected, and described in the following subsections. These descriptions are condensed versions of the full issue and recommended action descriptions contained in the "Guide for Multi-Architecture Live-Virtual-

Constructive Environment Engineering and Execution." As elaborated in the previous section, the issues for multi-architecture environments are not distributed evenly across DSEEP Steps; they are concentrated in the steps associated with designing, developing, and integrating/testing the a simulation environment. Thus examples have been chosen that address activities in these DSEEP steps.

4.1 STEP 3 EXAMPLES

The third step in the DSEEP is *Design Simulation Environment*. In this step, existing member applications that are suitable for reuse are identified, design activities for member application modifications and/or new member applications are performed, required functionalities are allocated to the member applications, and a plan is developed for development and implementation of the simulation environment.

ISSUE 1: *Object State Update Contents*

Some distributed simulation architectures (e.g., DIS) require updates of a simulated object's state to include a complete set of the object's state attributes. Other architectures (e.g., HLA) do not require object state updates to include attributes that have not changed. A multi-architecture simulation environment combining these two paradigms must resolve the difference.

RECOMMENDED ACTION(S)

The designer should ensure that the mechanisms used to link architectures with different state update requirements automatically produce updates that are compliant with the expectations of the receiving member applications. For example, DIS–HLA gateways typically perform these functions by maintaining a complete set of attributes for each simulated object [4] [5] [6]. When an HLA object attribute update for some object is received by the gateway, the gateway's internal attributes for the object are updated and then a complete DIS Entity State Protocol Data Unit (PDU) is produced from the gateway's internal attributes for the object and sent. When a DIS Entity State PDU for some object is received by the gateway, the object attributes in the incoming PDU are compared to the gateway's internal attributes for the object; those that are different are updated in the gateway's internal set from the PDU and also sent via an HLA object attribute update service invocation. The gateway's internal attributes for an object are initialized the first time the gateway receives an update for those attributes from either side.

ISSUE 2: *Object Identifier Uniqueness & Compatibility*

A common assumption in distributed simulation architectures is that data sent on the network describing the state of the simulated world is "ground truth," i.e., is correct with respect to that simulated environment. It is typically left to individual member applications to intentionally degrade or corrupt that ground-truth information in situations where a system or entity they are simulating would not have access to perfect and complete information. In some simulation environments, however, specialized member applications are used to perform that information degradation (e.g., modeling weather and terrain effects on communications in a communications effects server), and the degraded information is retransmitted on the network to other member applications. In multi-architecture simulation environments, such retransmitted degraded information must be translated from the originating architecture's data model and protocol into the other architecture's data model and protocol. This translation may occur in a gateway, in middleware, or elsewhere. The deliberately incorrect data may cause problems through its violation of the ground-truth assumption (e.g., seemingly inconsistent location data for a transmitting simulated entity) and the fact that information is transmitted twice (i.e., first in its original correct form and second in its degraded form).

RECOMMENDED ACTION(S)

This issue arises only if the deliberately incorrect non-ground-truth data is retransmitted in a form that is otherwise identical to the ground-truth data and thus indistinguishable from it, e.g., two correctly formatted DIS Entity State PDUs with different locations for the same simulated object. It is distinguishable from another Step 3 issue, Gateway Translation Paths, in that it arises not from inadvertent redundant translation in a gateway but from deliberate alteration and retransmission of data. One resolution to this issue is to use architecture features to distinguish ground-truth from non-ground-truth data. Such features could include different message types (e.g., a special HLA interaction class [7] [8]) or flags within a single message type. The translators (gateways, middleware) used to link the multi-architecture simulation environment must be able to correctly translate these non-ground-truth indicators into a form that conveys the same information after the translation.

If no suitable architecture/protocol features are available, it may suffice to modify the affected member applications to be aware of the sources of the different types of data (e.g., ground truth from the member application simulating an object, and non-ground truth from a communications effects server) and to use only

the incoming information from the desired source; this is an example of receiver-side filtering. Such modifications and intentions should be documented in advance in the simulation environment agreements (as part of DSEEP Activity 4.2: Establish Simulation Environment Agreements).

4.2 STEP 4 EXAMPLES

The fourth step in the DSEEP is *Develop Simulation Environment*. In this step, the simulation data exchange model is developed, simulation environment agreements are established, and new member applications and/or modifications to existing member applications are implemented.

ISSUE 1: *Meta-model Incompatibilities*

Differences in the underlying data exchange model structures used by different simulation architectures can cause incompatibilities in a multi-architecture environment. Specifically, the set of data fields that comprise an HLA Federation Object Model (FOM) (as specified in the HLA Object Model Template [OMT]), the set of fields that comprise a TENA Logical Range Object Model (LROM) (as specified in the TENA metamodel), and the set of fields that define DIS PDU structures are not the same.

Since the Simulation Data Exchange Models (SDEM) must align among the architectures in the multi-architecture environment, the team establishing the SDEM must understand these metamodel differences, understand the equivalencies and differences across the metamodels, and take actions so that each architecture's metamodel specifications are met in a consistent manner.

RECOMMENDED ACTION(S)

The fundamental representation of data in the metamodel of a given SDEM must be correlated across the architectures used in a given simulation environment. It is critical to the success of the simulation environment that the semantic meanings of the data representation in each SDEM be consistent. There are two recommended ways of resolving these incompatibilities: using an architecture neutral way of representing the metamodel and the use of gateways.

The ideal solution is to use an architecture neutral way of representing the metamodel. While there is on-going work to develop architecture neutral modeling mechanisms, as in the Joint Composable Object Model (JCOM) effort, it is most likely that the participating architectures model their attributes and behaviors in unique ways. Monitoring efforts such as JCOM and looking for opportunities to implement them into existing architectures is recommended.

The use of gateways is the primary recommended action for resolving existing metamodel incompatibilities. Several factors need to be addressed once the decision is made to use a gateway to link disparate architectures. The main questions to ask are: How do you choose the gateway? How do you know that the metamodel incompatibilities have been addressed? Is there a tool available to support gateway development across architectures?

Testing the gateway to ensure it is accurately translating the metamodel data is critical to the success of the simulation environment. Data on both sides of the gateway should be verified for syntactic and semantic accuracy. In addition to any tools provided by the gateway for data verification, architecture-specific data verification tools, if they exist, should be used to confirm or identify problems in the data translation.

ISSUE 2: *SDEM Content Incompatibilities*

Once the differences in metamodels is understood and addressed, the alignment of SDEMs across the multi-architecture environment must occur. The semantics of data to be exchanged must be understood, and equivalent across the architectures in use.

In some architectures, the metamodel and the content of the SDEM are defined as part of the architecture specification. While users typically enjoy the flexibility to tailor SDEM content to their immediate needs, it often comes at a price. Specifically, when working in a multi-architecture simulation environment, the wide spectrum of SDEM content across different architecture communities must be fully reconciled (within the context of the current application) if the various member applications are to interoperate correctly. This reconciliation process can be very expensive in terms of both time and resources, and can increase technical risk if not done correctly.

RECOMMENDED ACTION(S)

There are two recommended paths to explore when faced with SDEM content incompatibilities in a multi-architecture environment. First, member applications may be changed to support the native interface of a given architecture. Second, gateways may be used to bridge SDEM content incompatibilities. This could include the use of an architecture agnostic gateway/middleware solution. No matter which approach is taken, alignment must occur across the SDEMs: alignment of classes and class hierarchies, alignment of attribute assignments to

classes, and alignment of domains (including enumerations) to attributes.

Whether a change to the native interface of member applications or the use of a gateway is selected, there are tradeoffs to consider. How much will the SDEM force changes to the interfaces of each member application? What is the cost of using a gateway versus modifying the native interface? Sometimes this tradeoff will affect the selection of member applications to those that most closely align with the other member applications, within the same architecture and across architectures. Changing the native interface of member applications is usually the most expensive option of the three recommended approaches in both time and resources.

When using a gateway solution, time and resources should be spent to ensure the mappings in the gateway(s) are valid. In order for a gateway to accomplish its task, it is necessary to create detailed data mappings across architectures that specify the data level and data type exactly what will be passed through the gateway and how it will be represented on each side. This mapping becomes part of the documentation required to verify the correct operation of the gateway and also serve as a synchronization point between the architecture teams. When performing this manual mapping process the simulation environment developers should consider the following analysis types and "Measures of Similarity": morphological analysis, grammatical, semantic analysis, entity name similarity, entity descriptor name similarity, and semantic/usage similarity.

4.3 STEP 5 EXAMPLES

The fifth step in the DSEEP is *Integrate and Test Simulation Environment*. In this step, all necessary integration activities are performed, and testing is conducted to verify that interoperability requirements are being met.

ISSUE 1: *Multi-architecture Planning Considerations*

Multi-architecture development implies special consideration for execution planning beyond that normally required for a single architecture simulation environment. Failure to account for the additional complications of a multi-architecture simulation environment will result in an unrealistic execution plan.

RECOMMENDED ACTION(S)

The execution plan should address both the technical and soft (i.e., non-technical) factors associated with the operation of a multi-architecture environment. Examples of technical factors include the development of startup and shutdown procedures that are compatible with all of the architectures in use, and how to reconcile the different mechanisms used by the different architectures to pass large amounts of data over the simulation infrastructure. Examples of soft factors include how to train personnel to work with unfamiliar software and operational procedures, and the scheduling of personnel and facilities across users of multiple architectures. The federation agreements often provide a good basis for identifying the considerations that need to be addressed in an execution plan for a multi-architecture simulation environment.

ISSUE 2: *Initialization Sequencing and Synchronization*

Initialization in a distributed simulation is a non-trivial and sequential process. For example, in HLA, the federation execution must be created before federates can join and objects registered. Explicit sequencing and synchronization of the initialization actions of member applications in a simulation environment is frequently needed to ensure that each member application is ready for the next action in the initialization process. In a multi-architecture simulation environment, these issues may be exacerbated. Initialization sequencing needs may be greater because, for example, mechanisms used to link the multiple architectures, such as gateways or middleware, may require the architectures' executions to be started in a specific order. Such a sequencing constraint may be difficult to enforce. Moreover, explicit initialization synchronization may be more difficult in a multi-architecture simulation environment because the requisite synchronization mechanisms and messages (e.g., HLA synchronization services) are more likely to be architecture specific and less likely to be directly translatable across the architectures' protocols than more generic operations such as object attribute updates.

RECOMMENDED ACTION(S)

Some architectures offer protocol services that can be used, with varying degrees of effort, for initialization sequencing and synchronization. For example, synchronization points were used for this purpose in one HLA simulation environment, although they did not function as initially expected, necessitating a carefully planned common multi-phase initialization process that included planned pauses to allow initialization operations to complete [9].

If the protocol services of one architecture are used to coordinate initialization across architectures in a multi-architecture simulation environment, the mechanism used to link the architectures (such as a gateway or middleware) should be configured or modified to translate those services from one architecture to the

other. If synchronization services cannot be translated by the mechanism used to link the architectures, techniques outside of the simulation environment execution, such as manual control, may be used. Even if they can be translated, protocol services can only implement synchronization constraints that are known. To that end, specific attention should be given to initialization when planning the simulation environment execution and testing the simulation environment. Any synchronization constraints or initialization sequence decisions should have been documented in the simulation environment agreements. These initialization and synchronization services and procedures should be thoroughly tested at this point.

Depending on the needs of the specific simulation environment, software tools designed to monitor and control simulation environment execution (e.g., the TENA tools Starship and StarGen [10]) may be useful in sequencing and synchronizing initialization. Finally, to avoid these issues, member applications should be designed and implemented to be as independent of initialization sequence as possible [9].

5. NEXT STEPS

The DSEEP overlay for multi-architecture development and execution is intended as an open and continually available resource for the LVC community. Since the "parent" of this overlay is an existing standard, the logical next step for this product is to bring it forward for standardization. This action would allow for much broader exposure of the overlay elements to its intended user community, and through the participation of this community in the standardization effort, it is believed that the identification of issues will be more exhaustive and the associated recommended actions will better reflect current-day best practices in the LVC community.

Since the overlay explicitly references the various DSEEP activities and tasks, it follows that this close coupling should be also maintained at the standards level. Thus, because the DSEEP is a recognized IEEE standard, the DSEEP overlay will also be standardized underneath the IEEE. However, since the Simulation Interoperability Standards Organization (SISO) Standards Activity Committee (SAC) is a sponsor of IEEE simulation standards under the IEEE Computer Society Standards Activity Board (IEEE-CS SAB), the development of the standard will be performed by a SISO Product Development Group (PDG). A Product Nomination (PN) to establish the PDG was submitted to the SAC in June, and is expected to be approved well in advance of the 2010 Fall Simulation Interoperability Workshop (SIW) in Orlando FL. The first meeting of the DSEEP Multi-Architecture Overlay (DMAO) PDG is expected to be conducted at this SIW.

6. SUMMARY

This intent of this paper was to introduce the SISO/LVC community to the initial instantiation of a multi-architecture overlay to the DSEEP. This overlay was the result of an extensive literature search of various M&S archives to identify case studies of multi-architecture developments. The issues and recommended actions extracted from these case studies were augmented with the personal experiences of the technical team to provide substantial coverage of the special concerns associated with multi-architecture development. These issues and recommended actions were then aligned with DSEEP activities and tasks so that users understand where in the overall development process such concerns are encountered. The result is a common process view for multi-architecture development and execution that is equally relevant to all simulation architecture communities. This product will provide a solid foundation for upcoming standards activities under the IEEE.

7. REFERENCES

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

[2] Guide for Multi-Architecture Live-Virtual-Constructive Environment Engineering and Execution, The Johns Hopkins Applied Physics Laboratory, NSAD-R-2010-044, June 2010.

[3] Draft Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP), IEEE P1730, June 2010.

[4] Cox, A., D. D. Wood, M. D. Petty, and K. A. Juge. “Integrating DIS and SIMNET Simulations into HLA with a Gateway,” in *Proceedings of the 15th Workshop on Standards for the Interoperability of Defense Simulations*, Orlando, FL, 16–20 September 1996, pp. 517–525.

[5] Wood, D. D., M. D. Petty, A. Cox, R. C. Hofer, and S. M. Harkrider. “HLA Gateway Status and Future Plans,” in *Proceedings of the 1997 Spring Simulation Interoperability Workshop*, Orlando, FL, 3–7 March 1997, pp. 807–814.

[6] Wood, D. D., and M. D. Petty. “HLA Gateway 1999,” in *Proceedings of the Spring 1999 Simulation Interoperability Workshop*, Orlando, FL, 14–19 March 1999.

[7] Carr, F. H., and J. D. Roberts, "Incorporating Command and Control (C2) into the High Level Architecture (HLA): An Architecture for Realistic Communications Effects," in *Proceedings of the Fall 1997 Simulation Interoperability Workshop*, Orlando, FL, 8–12 September 1997.

[8] Lacetera, J., and E. Torres. "Evolution to a Common C4ISR M&S Environment via a Communications Effects Federate Approach," in *Proceedings of the Fall 1997 Simulation Interoperability Workshop*, Orlando, FL, 8–12 September 1997.

[9] Nielsen, J. "Federation Initialization and the RTI 1.3: Lessons from the JTLS-GCCS-NATO Federation," *Proceedings of the Spring 1999 Simulation Interoperability Workshop*, Orlando FL, March 14-19 1999.

[10] TENA Software Development Activity, "Advanced C2 Software Application, Starship II, Uses TENA For Distributed Test Environment", *TENA Fact Sheet*, On-line at https://www.tena-sda.org/display/intro/Documentation, Accessed March 27, 2010.

8. AUTHOR BIOGRAPHIES

ROBERT LUTZ is a principal staff scientist at The Johns Hopkins University Applied Physics Laboratory in Laurel MD. He has over 30 years of experience in the design, implementation, and evaluation of modeling and simulation (M&S) systems for military customers. Currently, he is leading the LVCAR "Systems Engineering Process" and "Common Gateways and Bridges" Phase II tasks, and supports 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. He also serves as a regular guest lecturer in The Johns Hopkins University Whiting School of Engineering.

ROY SCRUDDER is the Program Manager for the Modeling and Simulation Information Management Group at the University of Texas at Austin, Applied Research Laboratories (ARL:UT). His past experience includes serving as the Associate Director (Data) at the US DoD Modeling and Simulation Coordination Office (M&S CO). He has over 30 years' experience in information systems analysis and development, concentrating the last 18 years in information management for M&S. Mr. Scrudder's professional experiences are in the areas of data management and data engineering with a specialization in metadata. Mr. Scrudder's M&S standardization experience include serving as the Chair of the HLA Evolved Product Development Group and the Drafting Group Lead for the FEDEP. Mr. Scrudder holds a Bachelor of Science degree in Applied Mathematics from the University of Tennessee.

RONDA SYRING is a member of the Senior Professional Staff at The Johns Hopkins University Applied Physics Laboratory in Laurel MD. She holds a BS from the Naval Academy and MS from the Naval Postgraduate School in Monterey, CA. A retired naval officer, she served as an operations research analyst for OPNAV and OSD staffs. Currently she is an analyst supporting various projects and programs including providing cost analysis for Navy Aerial Targets and Decoy Systems Project Office, providing M&S support on airspace integration efforts for the Persistent Maritime Unmanned Aircraft Systems Program Office, and providing Navy manpower requirements analysis for the Bureau of Navy Medicine. She is an active member of the Military Operations Research Society.

MIKEL PETTY is Director of the University of Alabama in Huntsville's Center for Modeling, Simulation, and Analysis and a Research Professor in both the Computer Science and the Industrial and Systems Engineering and Engineering Management departments. Prior to joining UAH, he was Chief Scientist at Old Dominion University's Virginia Modeling, Analysis, and Simulation Center and Assistant Director at the University of Central Florida's Institute for Simulation and Training. He received a Ph.D. in Computer Science from the University of Central Florida in 1997. Dr. Petty has worked in modeling and simulation research and development since 1990 in areas that include simulation interoperability and composability, human behavior modeling, multi-resolution simulation, and applications of theory to simulation. He has published over 150 research papers and has been awarded over $13 million in research funding. He served on a National Research Council committee on modeling and simulation, is a Certified Modeling and Simulation Professional, and is an editor of the journals *SIMULATION* and *Journal of Defense Modeling and Simulation*. He was the dissertation advisor to the first two students to receive Ph.D.s in Modeling and Simulation at Old Dominion University.

JACK BORAH is the Chief Engineer for the USAF Air Force Modeling and Simulation Training Toolkit (AFMSTT) and a Senior Member of the Technical Staff for the AEgis Technologies Group. Currently, he is the Chair of the DSPT PRP and supports several M&S

standards initiatives within the Simulation Interoperability Standards Organization (SISO), including the IEEE 1730 Distributed Simulation Engineering and Execution Process (DSEEP) standard, the OMT component to the IEEE 1516 HLA standard, and the Simulation Conceptual Modeling Standing Study Group. He is a retired military officer with over 30 years experience in military training and training systems. He holds a BS from the Air Force Academy and MAS from Embry-Riddle Aeronautical University.

JOHN RUTLEDGE is a Senior Systems Architect with Trideum Corporation. Mr. Rutledge uses over 19 years of software and systems design, development and management experience in directly supporting the Joint Mission Environment Test Capability (JMETC) Program Office as an Event Support Lead. He served over 9 years in the United States Air Force (USAF) where he honed his skills in modeling and simulation development and design. Since leaving the USAF Mr. Rutledge has supported various projects from organizations including NASA, Missile Defense Agency, the US Army's Future Combat Systems (FCS) and now the Office of the Secretary of Defense.