Live Virtual Constructive Architecture Roadmap (LVCAR) Comparative Analysis of the Architectures

Robert Richbourg (IDA)
Robert Lutz (JHU APL)

September, 2008

Cleared for public release (Distribution A) by the DoD Office of Security Review
(Case No. 09-S-2416) / M&S CO Project No. 06OC-TR-001

List of Acronyms

ABCS	Army Battle Command System
ALSP	Aggregate Level Simulation Protocol
AMG	Architecture Management Group
AMT	Architecture Management Team
BLCSE	Battle Lab Collaborative Simulation Environment
BML	Battle Management Language
C2	Command and Control
C4ISR	Command, Control, Communications, Computer, Intelligence, Surveillance and Reconnaissance
CDD	Capability Development Document
COI	Community of Interest
CTIA	Common Training Instrumentation Architecture
DDM	Data Distribution Management
DDS	Data Distribution Services
DIS	Distributed Interactive Simulation
DMO	Distributed Mission Operations
DOD	Department of Defense
DODI	Department of Defense Instruction
DTRA	Defense Threat Reduction Agency
FEDEP	Federation Development and Execution Process
FOM	Federation Object Models
GIG	Global Information Grid
GTRI	Georgia Tech Research Institute
HLA	High Level Architecture
IDA	Institute for Defense Analyses
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
JBML	Joint Battle Management Language
JC3IEDM	Joint Command, Control and Consultation Information Exchange Data Model
JCIDS	Joint Capabilities Integration and Development System
JFCOM	Joint Forces Command
JHU APL	Johns Hopkins University Applied Physics Lab
JLVC	Joint Live Virtual Constructive
LVCAR	Live Virtual Constructive Architecture Roadmap
LVC-IA	Live Virtual Constructive Integrating Architecture
MOU	Memorandum of Understanding
M & S	Modeling and Simulation
MSG	Modeling and Simulation Group
MSDL	Mission Scenario Definition Language
NASM	National Air Space Model
NASMP	Naval Aviation Simulation Master Plan
NATO	North Atlantic Treaty Organization
NGA	National Geospatial-Intelligence Agency
NR-KPP	Net Read – Key Performance Parameter
OM	Object Model
OMG	Object Management Group
SAF	Semi-automated Forces
SEAS	Synthetic Environment for Analysis and Simulations
SEDRIS	Synthetic Environment Data Representation and Interchange Specification
SISO	Simulation Interoperability Standards Organization
SOA	Service Oriented Architecture
TENA	Test and Training Enabling Architecture

WG	Working Group

Table of Contents

List of Acronyms 2
Table of Contents 4
List of Figures 6
List of Tables 7
Executive Summary 8
1 Introduction and Overview 9
1.1 General Problem Statement 9
1.2 An Underlying and Fundamental Aspect of the Problem 10
1.3 Methodology 11
1.4 Levels of Interoperability 12
2 The Current State 13
2.1 Architecture Usage 13
2.2 Current Capability Requirements 14
2.3 “Wedge” Issues 17
2.3.1 Wedge #1: On-the Wire or Application Program Interface (API) Standard 18
2.3.2 Wedge #2: Single or Multiple Reference Frames 18
2.3.3 Wedge #3: Number of Compliance Levels 19
2.3.4 Wedge #4: Object Model Loading - Run-time or Compile time 19
2.3.5 Wedge #5: Global Event Ordering 20
2.3.6 Wedge #6: Standard Object Model Provided 20
2.3.7 Wedge #7: Object Model Extensible 21
2.3.8 Wedge # 8: Data Filtering Supported 22
2.3.9 Wedge #9: Reliable Transport Types 22
2.3.10 Wedge #10: Support Transfer of Ownership 23
2.3.11 Wedge #11: Data Marshalling Support 23
2.3.12 Wedge #12: Requirements for Tools / Utilities 24
2.3.13 “Wedge” Issue Summary 24
2.4 Projected Future Requirements 25
2.4.1 Improved Quality of Service (QoS) 25
2.4.2 Fault Tolerance 25
2.4.3 Information Assurance 25
2.4.4 Interface to GIG 26
2.4.5 Load Balancing 26
2.4.6 Projected Future Requirements Analysis Summary 26
2.5 Current State Analysis Summary 26
3 Defining the Desired End State 27
3.1 Conceptually or Physically Single Architecture 27
3.2 Established Levels of Service 27
3.3 Each DoD Context of Use 27
3.4 Allow Valid Interactions 27
4 Candidate Strategies 27
4.1 Strategy 1: Maintain the Status Quo 28
4.2 Strategy 2: Enhance Interoperability of Mixed-Architecture Events 28
4.3 Strategy 3: Encourage and Facilitate Architecture Convergence 29
4.4 Strategy 4: Select One of the Existing Architectures 29
4.5 Strategy 5: Develop a New Architecture 30
5 Assertions 31
6 Initial Strategy Assessment 35
7 Strategic Agility: Preserving Options is Desirable 37
8 General Problems for LVC Events 40
9 Creating Resources to Enhance Interoperability in Mixed-Architecture Environments 41
9.1 Common Components of Architecture-Independent Object Models 42
9.2 Produce Common Gateways and Bridges 43

9.3 Create a Common, Reusable Federation Agreement Template 44
9.4 Analyze, Plan, and Implement Improvements to the Processes and Infrastructure Supporting M&S Asset Reuse 44
9.5 Describe and Document a Common, Architecture-Independent Systems Engineering Process 45
9.6 Produce and Enable Production of Reusable Development Tools 45
9.7 Implement a Process to Maintain 'Living' Specifications of Current and Future Requirements 46
9.8 Specify a Resource or Capability to Facilitate Pre-integration Systems Readiness 47
9.9 Determine the Feasibility of a Common Wire-Level Protocol and a Common API 47
10 Convergence: Activities and Options 48
10.1 Activity: Convergence Technical Feasibility Determination 49
10.2 Activity: Architecture Convergence Plan 50
10.3 Initial Assessment of Convergence Targets 51
10.3.1 Required Capability Area: Publish and Subscribe Information Filtering 51
10.3.2 Required Capability Area: Transport Types 52
10.3.3 Required Capability Area: Interoperate with Other Protocols 52
10.3.4 Required Capability Area: Multiple Message Types 52
10.3.5 Required Capability Area: Support Save and Restore Operations 52
10.3.6 Required Capability Area: Support Region-based Information Management (filtering) 52
10.3.7 Required Capability Area: Support Ownership Transfer 52
10.3.8 Required Capability Area: Support Application Synchronization 53
10.3.9 Required Capability Area: Global Event Ordering 53
10.4 Summary: Initial Estimate of Feasible Areas for Architecture Convergence 53
11 Enabling Interoperability Now and Positioning for a Degree of Architecture Convergence in the Future 53
11.1 Graphical Representation of Activities and Influences 54
11.2 A More Complete Interoperability Problem 55
11.2.1 Design Pattern Analysis 56
11.2.2 Algorithmic Consistency 56
11.2.3 Environmental Representations 57
11.2.4 Human Behavior Modeling 57
11.2.5 Bridging Multiple Security Domains 57
12 Conclusions and Recommendations 57
12.1 Technical Area Conclusions 58
12.2 Technical Area Recommendations 58

List of Figures

Figure 2.1 Frequency of Cited Use (Percentage), by Architecture / Protocol 14
Figure 2.2 Architecture Capabilities: A High-level View 17
Figure 7.1 Strategy 3: Time, RRI, & Potential Course Correction Options 38
Figure 7.2 Strategy 2: Time, RRI, & Potential Course Correction Options 39

List of Tables

Table 2.1a Requirements Comparison 15
Table 2.1b Requirements Comparison (continued) 16
Table 10.1 Initial Estimate of Potential Areas for Capability Convergence 53

Executive Summary

Many problems exist with respect to the procedures and technologies used to develop mixed architecture live, virtual, and constructive (LVC) environments. The incompatibilities between these architectures require expending a considerable amount of resources to develop point solutions that effectively integrate them into a single, unified set of supporting simulation services. Gateway solutions to these types of issues have frequently restricted exercises to using only the limited set of capabilities that are common across all of the architectures, resulting in a “dumbing down” of the more capable architectures. Further, the lack of high-level management oversight of all existing distributed simulation architectures (as a unified resource) has resulted in a situation where continued 8ivergence of architectural capabilities is not only possible but likely, and new (potentially redundant) architectures can emerge at any time. Clearly, such issues must be satisfactorily resolved if long-term interoperability goals are to be achieved.

The architectures that were considered as part of this study include:

- Distributed Interactive Simulation (DIS)
- Aggregate-Level Simulation Protocol (ALSP)
- High-Level Architecture (HLA)
- Test and Training Enabling Architecture (TENA)
- Common Training and Instrumentation Architecture (CTIA)

Each of these architectures was designed to address the requirements of their defined user base. LVCAR Phase I efforts analyzed the core requirements of each architecture and directly compared key categories of requirements. There is a high degree of functional commonality between the architectures, particularly regarding HLA, TENA, and CTIA. However, there are also some key differences, stemming from specific needs that originally drove the development of each architecture.

At the implementation level, there are substantive differences among the architectures. Such differences are characterized as "wedge issues", potentially becoming barriers to achieving cross-architecture interoperability. The study finds that none of the wedge issues introduce irreconcilable incompatibilities that prevent the integration of the different architectures into mixed architecture events. However, achieving such integration is not without cost, and some degree of analysis/experimentation is required to determine the best near-, mid-, and long-term solutions to addressing these incompatibilities.

A set of five potential strategies has been identified, and a corresponding set of advantages and disadvantages associated with each of them. An analysis of these factors, along with an assessment of how well each strategy met requirements derived from assertions that characterize the current LVC interoperability picture, led to the elimination of three strategies from further consideration. The remaining strategies (Strategies 2 and 3) are defined as follows:

- Strategy 2: Define and develop mechanisms to improve LVC interoperability in mixed-architecture environments, assuming that the current architectures will continue to be used.
- Strategy 3: Develop policy and incentives to encourage existing architectures to converge (over some defined period of time) to either a single architecture or a smaller set of architectures.

The final phase (Phase II) of the LVCAR study effort resulted in a more detailed assessment of the two remaining strategies. Phase II efforts also developed the temporally sequenced set of actions required to implement the strategy. Each action was assessed according to its inherent benefit and cost, and value considerations strongly influenced the final configuration of the LVC architecture roadmap.

1 Introduction and Overview

The goal of the LVCAR study is "to methodically and objectively develop a recommended roadmap (way forward) regarding LVC interoperability across three broad areas of concern: definition of the desired future integrating architecture(s), the desired business model(s), and the manner in which standards should be evolved and compliance evaluated. The roadmap will also provide:

- Rationale for these recommendations, citing the findings on which they are based
- An assessment of how any LVC architecture policy change might be perceived in the user communities, with a recommendation on optimal ways to communicate this new direction
- Recommended next steps (e.g., experiments, prototyping new architecture(s), etc.)"

Sponsored and funded by the U.S. Department of Defense (DoD) Modeling and Simulation (M&S) Steering Committee (SC), the study will explore and assess a number of alternatives supporting simulation interoperability (at the technical level), business models, and the evolution process of standards management across the Department. The goal is to define an efficient, effective path to maximize technical interoperability of M&S systems across the U.S. DoD. This document provides the findings developed by the LVCAR Architecture team.

1.1 General Problem Statement

The early 1990's saw many accomplishments in the area of distributed simulation. The earliest successes came through the SIMNET program, which demonstrated that geographically dispersed simulation systems could support distributed training by interacting with each other across network connections. Following on this success, the Aggregate-level Simulation Protocol (ALSP) extended the benefits of distributed simulation to the force-level training community so that different aggregate-level simulations could cooperate to provide theater-level experiences for battle-staff training. At about the same time, the SIMNET protocol evolved and matured into the Distributed Interactive Simulation (DIS) standard. DIS allowed an increased number of simulation types to interact in distributed events, but was again focused on the platform-level training community. In the middle 1990's, the Defense Modeling and Simulation Office (DMSO) started the High-Level Architecture (HLA) program to combine the best features of DIS and ALSP into a single architecture that could also support uses in the analysis and acquisition communities, while continuing to support training applications. HLA was designed from the start to support a wide set of potential user communities.

As is the case with most broadly applicable tools, some began to perceive HLA as a "jack of all trades, but master of none." In particular, two communities started development of alternate architectures due to their perception that HLA included unacceptable performance limitations. The real-time test range community started development of the Test and Training Enabling Architecture (TENA) to provide low-latency, high-performance service in the hard-real-time application of integrating live assets in the test-range setting. Similarly, the Army started development of the Common Training Instrumentation Architecture (CTIA) to interconnect live assets on Army training ranges as they participated in large-scale exercises.

With the exception of SIMNET, all of these architectures remain in service today. Some are in early and growing use (e.g., CTIA, TENA) while others have seen a reduction, but not elimination, of their user base (e.g., ALSP). Each of the architectures is providing an acceptable level of capability within the areas where they have been adopted. However, ALSP, DIS, HLA, TENA, and CTIA federations are not inherently interoperable with each other. Thus, when simulation events include applications that rely on the different architectures, additional steps must be taken to allow effective communication between all applications. These additional steps, typically involving interposing gateways or bridges between the various architectures, may introduce increased risk, complexity, cost, level of effort, and preparation time into the event. Additional problems extend beyond the implementation of individual simulation events. As a single example, the ability to reuse supporting

models, personnel (expertise), and applications across the different protocols is limited. In short, the limited inherent interoperability between the different protocols introduces a significant and largely unnecessary barrier to the integration of live, virtual, and constructive simulations. This barrier must be greatly reduced or eliminated.

1.2 An Underlying and Fundamental Aspect of the Problem

There is a perception by many in the LVC community that interoperability will be much easier (and less costly) if there is only a single architecture available for use. Stated somewhat differently, many share the opinion that reducing the number of available architectures to one would result in a far easier interoperability situation than we face today. Included in this perception is the concept that the Department would benefit by eliminating the costs associated with maintaining overlapping capabilities. The desire to achieve a single-architecture state is based on a number of difficulties in the current situation that can be directly attributed to the existence of multiple architectures.

First, problems arise whenever multiple architectures must be integrated for use in a single event. In many cases, such mixed-architecture events can only use the set of capabilities common across all of the architectures to be included in the event. This is sometimes described as the "dumbing down" of the more capable architectures because the full range of unique capabilities they offer (e.g., more advanced capabilities such as repeatability, communications bandwidth efficiency, ownership transfer) cannot be used across the entire set of participating systems. Further, the costs required to integrate architectures rarely contribute directly to achieving simulation event goals. Instead, the associated costs usually provide point solutions, versions of which have likely been created in the past and probably will be paid for again in the future. Thus, the integration costs are viewed as recurring expenses that contribute little to achieving event goals and should thus be eliminated. Mixed architectures also impede "plug-and-play"; it is slower, more expensive, and sometimes impractical to compose simulation events using any of the wide range of assets (e.g., simulations, simulators, labs, ranges, C4ISR systems, etc.) available in the DoD inventory. In this view, one cannot simply choose an asset based on functional merit alone in all cases; frequently, the asset also is constrained to be compatible with a specific architecture. If the "out-of-the-box" compatibility constraints are ignored, some amount of additional cost (time, dollars, etc.) often follows. Typically, this cost cannot be ignored, so event designers will not even consider incompatible assets and will thus design events without having a complete picture of the resources that might have been used.

However, while each of these disadvantages can be attributed to the existence and simulation system use of multiple architectures, their existence does not necessarily justify an assumption that ridding the DoD of all but one architecture would result in an optimal state of affairs. In light of past experience, there are at least five main factors indicating such an assumption appears to be fallacious. First, legacy systems will continue to be used and it is unlikely that these systems will upgrade to using a new or different architecture. Thus, use of legacy systems is most likely to preclude the possibility of ever achieving a truly "single-architecture" state. Second, use of a single architecture may still require the use of supporting bridges, much as use of different RTIs (all derived from the same HLA architectural specification) can require bridges today. Third, gateways will be required for connecting any single simulation architecture to C4I systems, to the GIG, or, in general, to any type of system that has a primary purpose outside of the simulation arena. Fourth, the alignment of a family of simulations on a single architecture represents a single point solution. Having attained such standardization, history points to the likelihood that the diverse group of simulation users will quickly diverge into specializations, leading to the need for gateways to bridge their differences. Fifth, the selection or creation of a single architecture assumes that the rapid advances of the commercial software industry will not lead to a better implementation in the future, perhaps based on a Service-Oriented Architecture (SOA) paradigm. When this does occur, the existing standard architecture would be abandoned by users who have needs for the superior architecture delivered by the commercial sources.

The simultaneous existence of multiple architectures may allow benefits that are less likely to be achieved in a single architecture state. These include: 1) the ability to support multiple business and standards-use communities simultaneously and; 2) fostering the capability to "use the right tool for the job", avoiding the "one size fits all" problem. Some specific examples include:

- DIS: This protocol has a comparatively low barrier to entry; it is relatively simple to learn and easy to use. Also, it imposes a very low overhead. Whenever simulation events do not require using more advanced architectural services (such as time management, region-based information filtering, and so on), DIS offers a very economical solution to the system intercommunication problem.

- HLA: This architecture can serve a disparate collection of simulation systems, including those that require advanced architectural services and those that have modest requirements. In addition to its large U.S. user base, it's standing as an international standard has resulted in a large level of use in the coalition partner countries, facilitating combined simulation events that include multiple nations.

- TENA: This is a very capable architecture, offering much of the same capability as HLA, but based on more modern object-oriented technology. TENA middleware is offered to government users as GOTS, unlike the HLA that must be purchased on a per-seat basis.

- CTIA: This architecture uses the service-oriented paradigm and is unique in that respect. Also, it has been designed to continue providing some level of service even in the face of unreliable communication networks. It also provides advanced service capabilities while providing an "on-the-wire" specification (instead of an API-level standard), thus offering potentially improved support for multiple hardware platforms, operating systems, and software development languages.

In short, the existence of multiple architectures is not necessarily an undesirable outcome and, given some of the unique benefits, could be a desirable outcome if the architectures can be easily integrated.

In summary, there are advantages and disadvantages associated with the number of architectures that are available for use. There is no paramount advantage or disadvantage that allows one to immediately recognize the best possible solution. A significant problem for the LVCAR roadmap effort is to navigate this trade space to arrive at an achievable solution that maximizes the benefit for all concerned while not exceeding the resources that will be necessary to realize that solution.

1.3 Methodology

The original specification of requirements for the LVC Architecture Roadmap effort includes as a deliverable: "A systems engineering analysis of alternative means to achieve LVC architecture interoperability or convergence, considering all relevant factors, …, and providing metrics as feasible. Innovative approaches shall be considered." As explained by the International Council on Systems Engineering in the INCOSE Handbook, "The basic engine for systems engineering is an iterative process that expands on the common sense strategy of (1) understanding a problem before you attempt to solve it, (2) examining alternative solutions (do not jump to a point design), and (3) verify that the selected solution is correct before continuing the definition activities or proceeding to the next problem."[1] There are many formally-established and recognized models that help engineers execute this process, including the various Waterfall models, the Spiral model, and well-regarded "Vee" model of systems development. The common factor in each of these models is that they focus early on defining and documenting needs and requirements as a precursor to formal problem definition, development of a solution (to include the development and analysis of alternatives), and finally implementation and validation of the selected course of action. The LVC Architecture Roadmap is

[1] http://g2sebok.incose.org/app/mss/menu/index.cfm, Section 2.2.1, "INCOSE Handbook SE process Model".

concerned with the first steps in this process and is not chartered to address implementation and validation issues.

Given that "… that there is no cook-book recipe for performing systems engineering because there is such great diversity in the kinds of systems that are produced by the systems engineering activities …"[2], the architecture team approach to the systems engineering process for this problem is designed to focus on requirements and the development of alternative solutions that meet those requirements. Our process is to: 1) characterize the current state; 2) define a desirable end state; 3) identify differences between the current and desired state and develop strategies that offer promise in eliminating the differences, and; 4) perform an analysis of alternatives to select the most favorable course of action. Each of these steps is described more fully below. Generally, steps 1 and 2 correspond to the first step in the INCOSE common-sense strategy of understanding the problem. Similarly, the final two steps allow us to "examine the alternative solutions."

A characterization of the current state (see Section 2) has been developed based on literature reviews, subject-matter expert interviews, workshop interactions, survey instruments, and expert team input. All of these input sources have contributed to an identification of the architectures currently in use, their degree of use, and a description of the types of use for each (e.g., by community of interest and as stand-alone resources or as part of mixed-architecture events). Concurrently, the characterization of the current state has also led to a definition of the current architectural requirements. The study team has captured a list of key issues ("wedge issues") which provide the basis for a community perception of differences between the architectures. The current state characterization also includes analyses of how the requirements have been met and the actual incompatibilities of the architectures, stemming from the wedge issues. Finally, the current state analysis has also led to a definition of likely future requirements.

The vision statement (see section 3) describing the desired end state has been derived from the goal of establishing the seamless integration of live, virtual, and constructive systems. The same information sources used to characterize the current state (e.g., survey instruments, expert team input, etc.) have also helped to shape this vision. Further, the desired end state has been decomposed to permit a more detailed description of its associated attributes

The development of alternative strategies to transition from the current state to the desired end state has been inspired by the INCOSE principle of "innovate by generating a wide range of alternative solutions to satisfy the need. (A common mistake is to converge on a "comfortable design" concept too early because of time constraints.)"[3] Five different strategies have been proposed as potential solutions. Some of the solutions in this initial set are mutually exclusive while others are more flexible and fully support the INCOSE principle of repeating analyses to (re) investigate the desirability of the alternative solutions as better data becomes available.[4] Finally, a part of the alternative solution development has been to document a set of relative advantages and disadvantages ("pros and cons") of each one.

1.4 Levels of Interoperability

Many of the known problems that impact LVC integration stem from technical incompatibilities among the various distributed simulation architectures. However, achieving the goal of a truly interoperable LVC operating environment requires that developers consider a wide range of issues beyond the basic question of how to pass runtime data along the simulation network. As examples, issues related to data modeling, coordinate systems, synthetic natural environment representation, and algorithmic consistency are frequently outside the scope of the problems that simulation architectures

[2] http://g2sebok.incose.org/app/mss/menu/index.cfm, Section 2.2.0, "Overview"
[3] http://g2sebok.incose.org/app/mss/menu/index.cfm, Section 2.1.4.4, "Identify and Assess Alternatives so as to Converge on a Solution"
[4] http://g2sebok.incose.org/app/mss/menu/index.cfm, Section 2.1.2.2, "Investigate Alternatives"

were ever designed to address. The consideration of such issues as part of a structured systems engineering methodology is critical if executions of the LVC environment are to produce valid results.

The following framework is one tool useful in assessing interoperability (although others have also been proposed). It identifies five basic levels of simulation interoperability, with each level building upon each preceding level[5]:

Technical Interoperability (*Level 1*): Physical connectivity is established, allowing bits and bytes to be exchanged.

Syntactic Interoperability (*Level 2*): Data can be exchanged in standardized formats (i.e., the same protocols and formats are supported.

Semantic Interoperability (*Level 3*): Information (data plus context) can be exchanged. Common reference models define the unambiguous meaning of data.

Pragmatic/Dynamic Interoperability (*Level 4*): Knowledge (information plus its use and applicability) can be exchanged.

Conceptual Interoperability (*Level 5*): A common view of the world is established. This level not only comprises the implemented knowledge, but also the interrelations between these elements.

LVC interoperability is affected across all of these levels. The distributed simulation architectures in use within the DoD today all provide services for achieving technical and syntactic interoperability (e.g., levels 1 and 2); however, problems with how these services interact at runtime can adversely affect interoperability in mixed architecture environments. While solutions can be found to such problems, it is important to recognize that most practical distributed simulation applications require interoperability at levels above the syntactic level. Addressing interoperability issues at the semantic level (and above) frequently transcends the architectures themselves, and generally involves the establishment of cross-community agreements and standards on such supporting resources as data, processes, and tools. Thus, although the primary focus of the LVCAR is on the syntactic-level issues of mixed architecture integration, the general desire to reduce the technical, cost, and schedule risks associated with developing and operating future LVC environments also requires the consideration of higher-level interoperability issues. This will be discussed later in this report.

2 The Current State

An assessment and description of the current state has been developed based on literature reviews, subject-matter expert interviews, workshop interactions, survey instruments, and expert team input. The current state attributes of interest are: 1) the architectures currently in use; 2) the requirements that these architectures have been designed to address; 3) differences between the implementations that meet those requirements, and; 4) potential future requirements. Each of these areas is described in the sections that follow.

2.1 Architecture Usage

As described in the project plan, the currently employed architectures and protocols include the Distributed Interactive Simulation (DIS) protocol, the Aggregate-Level Simulation Protocol (ALSP), the High-Level Architecture (HLA), the Test and Training Enabling Architecture (TENA), and the Common Training and Instrumentation Architecture (CTIA). Each of these has an established community of use, and the number of users is increasing in each case except ALSP (where the user base appears

[5] Tolk, A., "Composable Missions Spaces and M&S Repositories – Applicability of Open Standards", Simulation Interoperability Workshop, Washington D.C., April 2004.

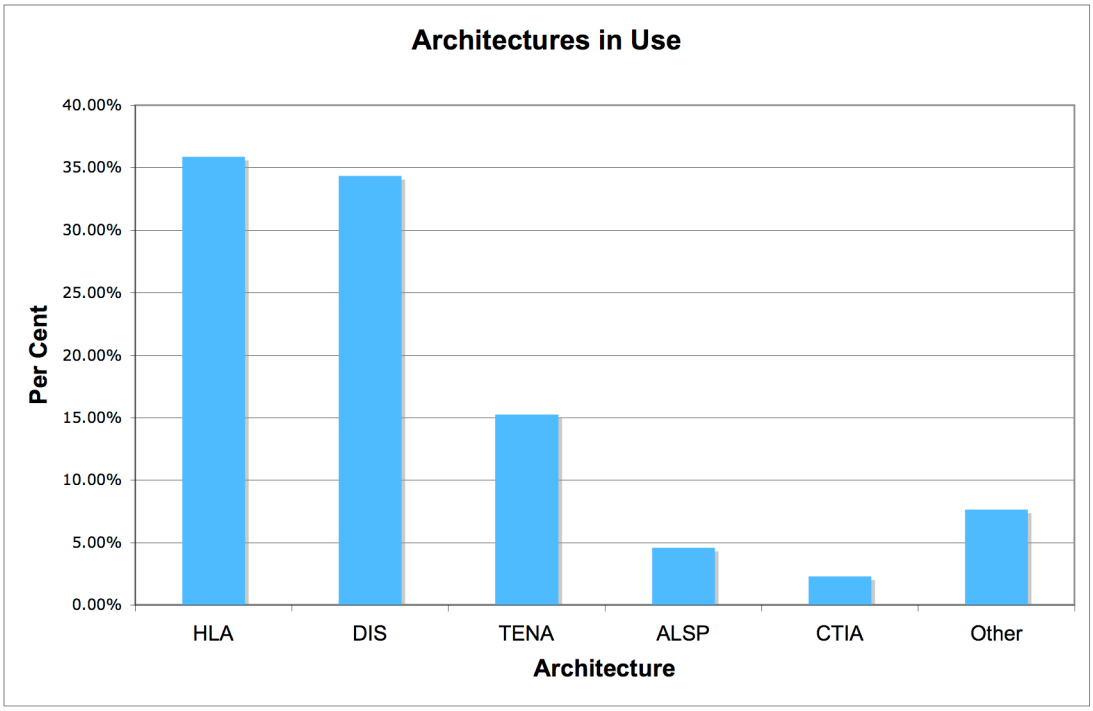

Figure 2.1 Frequency of Cited Use (Percentage), by Architecture / Protocol

to be vanishing as users migrate to HLA). Figure 2.1 is based on the LVCAR initial survey instrument completed in August 2007. Respondents were allowed to cite use of multiple architectures; the percentages above are based on the total number of cited use (so that sum of all cites uses equals 100%). Survey data shows that HLA and DIS have the largest frequency of cited use and together account for seventy percent of all simulation uses within the population represented by the survey audience. Detailed demographic information on the survey audience is presented elsewhere in the LVCAR Final Report.

Survey completion was a requirement to participate in the LVCAR working group. There were 105 respondents contributing to the data in Figure 2.1. While not a large number, the respondents represented the majority of the Department of Defense simulation communities. The "Other" category in the data includes several proprietary and custom-built architectures that are not generally available for public use. (Note that some survey respondents cite use of multiple architectures.)

2.2 Current Capability Requirements

A companion LVCAR report provides greater detail on the requirements collection methodology and results. This section provides a comparative presentation of the requirements collection effort, illustrating the degree of commonality between the various architectures and protocols. In the tables that follow, entries colored green indicate a high degree of commonality, yellow indicates a moderate degree of commonality and red indicates significant differences in the requirements. Note that definitions associated with the categories used in this comparison are provided in Appendix L (Definitions) of the LVCAR Interim Report.

Requirement	DIS	HLA	TENA	CTIA	ALSP
Create a distributed simulation, allow systems to join &	Limited. Simulation management	Yes	Yes	Yes	Yes

resign; provide for initialization and destruction of the distributed simulation instance	requirements reflected in *Simulation Management* (SIMAN) Protocols				
Support publish and subscribe information management (filtering)	No. However, the DIS architecture must support either multicast, broadcast, or unicast packets	Dynamic	Dynamic	Dynamic	Dynamic
Transport Type	Only supports best effort messaging	Support both best effort and reliable messaging	Support both best effort and reliable messaging	Supports reliable and unreliable networks. UDP unicast and multicast protocols supported	Supports reliable messaging
Interoperate with HLA federations	No requirement, although HLA did not exist when the DIS specifications were written. DIS gateways are now plentiful		"One of the primary goals of TENA is to interoperate with HLA simulations, therefore a TENA-to-HLA gateway is one of the many gateways that is needed." See 3.6, "Foundation Initiative 2010, Second TENA Middleware Prototype Requirements Document"	Yes - CTIA has a requirement to provide gateways to non CTIA-compliant protocols and architectures such as DIS and HLA	No requirement
Support multiple message types	Limited. Simulation management requirements reflected in *Simulation Management* (SIMAN) Protocols	Supports interactions and state updates	Supports state updates, single messages, and data streams	Yes: Supports state updates, single messages, and data streams	Supports state updates, single messages, and data streams

Table 2.1a Requirements Comparison

Requirement	DIS	HLA	TENA	CTIA	ALSP
Save and restore operation	Yes: Provided by SIMAN protocols	Yes	Original Requirement: "… middleware support for application save and restore is not necessary in a range context …" TENA requirements have been updated to reflect a need to provide this capability	Yes: Exercise is continuously saved (all exercise data is logged to a centralized DBMS)	Yes for Save. Spec doesn't discuss Restore, but it can be safely assumed
Region-based information management (filtering)	No Requirement	Yes	Original Requirement: No. TENA requirements have been updated and TENA will be enhanced to provide a DDM-like capability with the next release (Nov 07)	Yes: Discrimination by object class, attribute value / range, and geographic position	Yes. Discrimination by object class, attribute value/range, and geographic location (e.g., play box)
Transfer of ownership	Object Transfer	Attribute Transfer (Object transfer through privilegeToDelete attribute)	Object Transfer	Objects are centrally owned, but can transfer the right to change object state	Dynamic attribute transfer
Synchronize applications	At Stop/Freeze points; See SIMAN protocols	At any time	At Initialization: "TENA middleware shall not be responsible for ensuring that system clocks are synchronized."	No: Only synchronization through operating system capabilities	At Save points
Global Event Ordering	No. However, time management options shall include real-world time, scaled real-world time, and scaled/stepped real-world time	Both: Time-stamped order and Received order supported	"TENA applications need to be able to deal with Time Management issues only when connecting to simulations via the HLA Gateway. Thus, no specific time management requirements exist for the TENA middleware	Events are time-stamped when generated but may be delivered in arbitrary order. Subscribers must be capable of handling events out of order. However, all events are logged to the event database to preserve event ordering for AAR.	Yes: always
Specification for Middleware, Tools & Utilities	No requirement	Very Limited: RTI	Yes: Data Logger, Object Model, Middleware, … A "Product Line" Architecture	Yes: Extensive "product line" requirements for a large number of tools & utilities	Limited: ALSP Broadcast Emulator (ABE), ALSP Common Module (ACM), …

Table 2.1b Requirements Comparison (continued)

The comparative analysis in the above tables illustrates that there is a high degree of commonality between the architectures and protocols. This is particularly true concerning HLA, TENA, and CTIA. However, there are also some key differences that have been indicated in the specifications of requirement for these architectures. As an example, CTIA has a documented requirement to provide a persistent database of all objects; the architecture is required to support persistence of component identities across restarts. Thus, all information is continually recorded in an SQL-type database. This requirement is different than that imposed on a data logger which records interactions sent across the simulation network. CTIA is the only architecture that supports such a requirement (although TENA has been required to "support the local collection of data to a persistent store"). CTIA is also uniquely required to support continuation of exercises in cases where the transport network itself is unreliable.

TENA has also been presented with a unique requirement that is not technical. The TENA JORD states: "SW-6: the software must minimize the purchase of run-time licenses." This appears to be the only example where a purchase requirement has been expressed as an architectural requirement.

The considerable amount of capability overlap (considering only major characteristics), as well as some of the aspects unique to each of the architectures is illustrated in Figure 2.2.

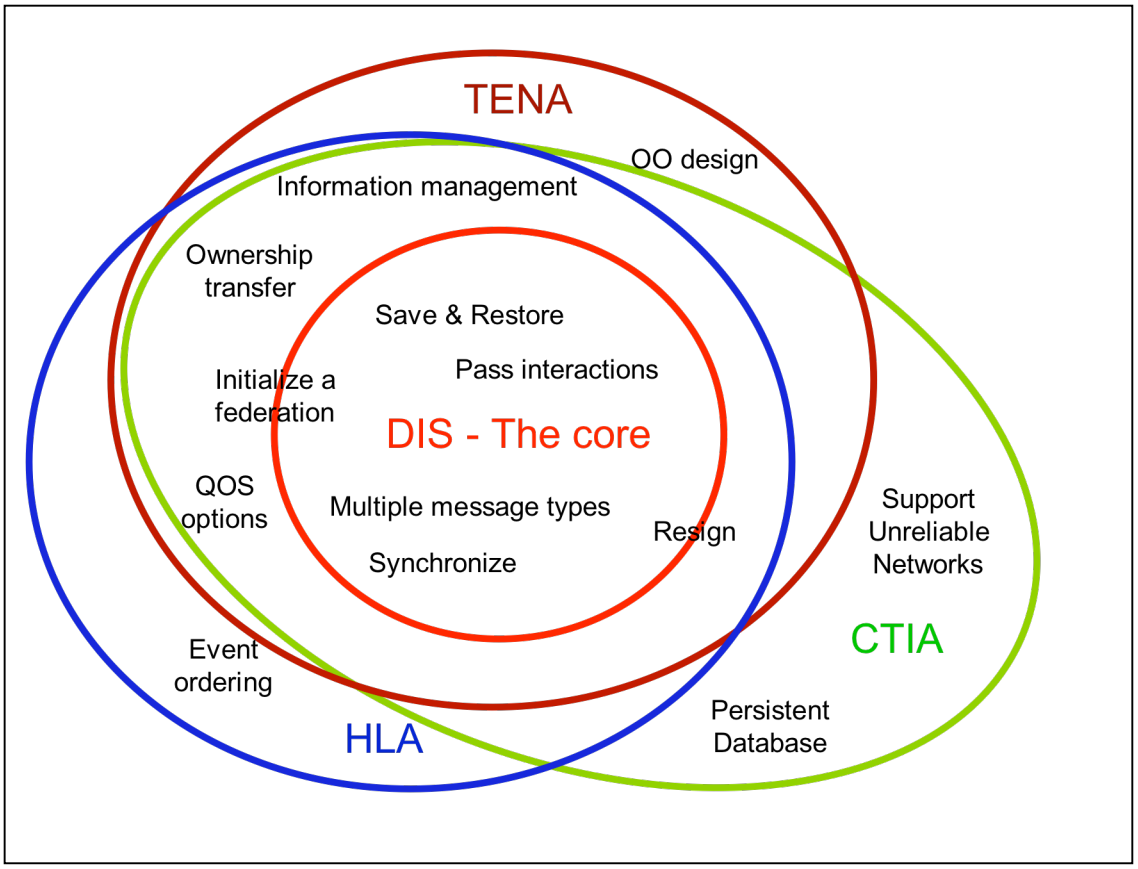

Figure 2.2 Architecture Capabilities: A High-level View

2.3 "Wedge" Issues

"Wedge" issues are characteristics of the different architectures and protocols that result from the different implementations of their requirements. These implementation characteristics are popularly seen as "wedges" between the architectures because they represent areas that could be barriers to achieving cross-architecture interoperability. This section examines the relevant wedge issues,

describes the implementation choices made in the case of each architecture or protocol, and describes the potential to resolve incompatibilities of the wedge issues. (See Appendix L for related definitions.)

2.3.1 Wedge #1: On-the Wire or Application Program Interface (API) Standard

DIS	**On-The-Wire**: An on-the-wire protocol enforces data structure/encoding rules, and thus strongly facilitates syntactic interoperability. Since the DIS community has a well-defined set of users (real-time, platform-level only), it was believed that long-lived agreements on the wire protocol could be obtained.
HLA	**API Standard**: On-the-wire specifications can require frequent updates to account for technological advances in how data is transmitted. Since HLA is an established standard (normally updated at five year cycles), making frequent updates would be very disruptive to users. An API standard hides the details of how data transmission takes place, and thus allows new technologies to be integrated without affecting the core standard.
TENA	**API Standard**: On-the-wire specifications can require frequent updates to account for technological advances in how data is transmitted. Having to provide a new version of TENA every time a new data transmission technology is used would be very disruptive to users. An API standard hides the details of how data transmission takes place, and thus allows new technologies to be integrated without affecting the core standard.
CTIA	**On-The-Wire**: CTIA chose an on-the-wire specification to support development of components on multiple operating systems in virtually any language, without requiring a specific middleware solution. This allows independent development of components for virtually any operating system, hardware platform and in any programming language. This also allows CTIA to optimize the interfaces to maximize performance over limited bandwidth networks (i.e. wireless).
ALSP	**On-The-Wire**: ALSP chose an on-the-wire specification to avoid the need to change the using simulations in any way (no code changes to the federates) - this had the beneficial side-effect of insulating the infrastructure from the simulations - when one simulation failed (crashed) the communications infrastructure remained fully operational and could help restore the state of the failed federate when possible.

There are advantages and disadvantages to both approaches. However, translation to and from on-the-wire standards can be handled very efficiently at the gateway level, and can be used to achieve interoperability between the architectures. This wedge issue does not appear to introduce irreconcilable differences between the different architectures.

2.3.2 Wedge #2: Single or Multiple Reference Frames

DIS	**Single**: A single coordinate system used through the connected systems can improve efficiency and resolve some semantic issues. However, if transformations to the single in-use systems are calculated differently (e.g., using different datum) inconsistency can result.
HLA	**Single**: (same discussion as above for DIS)
TENA	**Multiple**: Using multiple, 'native' coordinate systems eases application burden and a single specification of the transformation method increases the probability of coordinate consistency. TENA provides services from the SEDRIS Spatial Reference Model (SRM) to achieve coordinate transformation.
CTIA	**Single**: CTIA selected WGS84, Earth-Centered, Earth-Fixed reference frame, with a local range offset applied (to provide high precision while reducing required bandwidth). Components that use the CTIA Frameworks utilize the Synthetic Environment Data Representation and Interchange Specification (SEDRIS) coordinate conversion utilities to ensure consistent coordinate transformations.
ALSP	**Single**: A single coordinate system used through the connected systems can improve

	efficiency and resolve some semantic issues. However, if transformations to the single in-use systems are calculated differently (e.g., using different datum) inconsistency can result.

This is a minor consideration in architecture design and requirements specification. Either approach can lead to inconsistencies, depending on the implementation and the choices in the transformation process. Translation to and from different coordinate systems can be handled very efficiently at the gateway level, and can be used to achieve interoperability between the architectures. This wedge issue does not appear to introduce irreconcilable differences between the different architectures.

2.3.3 Wedge #3: Number of Compliance Levels

DIS	**None**: No compliance testing required to claim being a DIS-compliant system.
HLA	**Single**: Compliance for HLA federate simulations is entirely driven by the federate Simulation Object Model (SOM). The SOM defines the information that the simulation can produce and consume, and the services it can support. Thus, compliance testing is always done within the context of a simulation's SOM. Although this means that a simulation can define a minimal interface and still declare itself "HLA compliant", the fact that its SOM reflects very minimal capabilities and thus will be unlikely to be chosen for a federation is a strong deterrent. (Note that HLA also provides compliance testing for the RTI.)
TENA	**Multiple**: TENA developed several levels of compliance, all focused on the degree to which each simulation conforms to the rules of architecture. By defining multiple levels, it is more straightforward to identify and integrate compatible simulations, and potential users have a greater understanding of the capabilities of a given simulation with respect to its ability to operate in a TENA environment.
CTIA	**Multiple**: CTIA defines multiple levels of compliance, all focused on the degree of reusability for components. Higher levels of compliance indicate greater potential for reusability.
ALSP	**Double**: ALSP has two basic levels of "good actor" testing. One level is to ensure simulations can consume (subscribe to) information as defined by the ALSP specification. All ALSP actors must be able to support this for the confederation to operate effectively. An additional set of compliance tests are performed on actors that can also publish information. These two levels are necessary to avoid subscribe-only actors being submitted to unnecessary tests.

This is a minor consideration in assessing architecture compatibility. Gateways are not required based on the differences. Multiple levels of compliance could be used with HLA federations, but are somewhat irrelevant given the absence of a standard object model. Also note that TENA design requirement RM-2 states: “As TENA-compliant, the Resource Manager shall also be compliant with HLA and be able to manage HLA-compliant resources.” This wedge issue does not appear to introduce irreconcilable differences between the different architectures.

2.3.4 Wedge #4: Object Model Loading - Run-time or Compile time

DIS	**N/A**: DIS does not have an object model
HLA	**Run-time**: Permits dynamic changes to the object model (Modular FOM) and does not require recompile subsequent to object model changes.
TENA	**Compile-time**: Compiler error checking is applied to the object model (e.g., data type consistency) so that some errors can be discovered prior to run-time integration.
CTIA	**Both**: The majority of the CTIA Object Model is specified at compile-time, which facilitates compiler error checking. However, a portion of the CTIA Object Model is specified at run-time, to support flexibility, where required, in the definition of the Object Model for a particular range, or component.

ALSP	**Run-time**: Although the ALSP object model was generally fixed, developers wanted the flexibility of entering an arbitrary object model at runtime rather than compiling the object model into their code.

This is a minor consideration in assessing architecture compatibility and is most closely related to initialization procedures. Gateways are not required based on the differences. This is a major consideration in individual architecture design, having both technical and business implications, but this wedge issue does not appear to introduce irreconcilable differences between the different architectures.

2.3.5 Wedge #5: Global Event Ordering

DIS	**No:** DIS is for real-time platform-level applications where some mismatches in event order are tolerable due to human delays in recognizing and reacting to such mismatches.
HLA	**Yes:** HLA is a general-purpose simulation architecture that has a hard requirement to support both FTRT and STRT applications that require strict causal ordering of events. Although this is only an issue with constructive federations, there is a sufficiently large user base to justify that appropriate event ordering and time synchronization mechanisms are supported by the architecture.
TENA	**No:** At this time, all TENA applications are real-time, for which minor event ordering mismatches are tolerable.
CTIA	**No:** At this time, all CTIA applications are real-time, for which minor event ordering mismatches are tolerable. CTIA does provide event ordering for historical queries (a persistent store of exercise history is included as part of the architecture) and CTIA supports recovery of events (i.e. due to loss of communications) with special handling.
ALSP	**Yes**: ALSP is a constructive, time-managed simulation environment that depends on the ability to accurately order events within each time step (generally one minute).

This difference does not seem easily reconcilable. Gateways can be used to change time-management approaches as necessary, but can only time-stamp a message when that message is received at the gateway. If messages become out-of-order before arriving at the gateway, the gateway cannot restore the correct order. The problem can be resolved to a large degree (probably perfect resolution is possible) by interposing a “time-stamping” gateway between each message-generating application and the simulation network (as opposed to the conventional use of gateways between different architectures). This strategy would likely require many more gateways to be used than the conventional gateway-use strategy and thus could impose performance costs on the simulation exercise. However, it is not impossible to resolve the differences between time-managed and non-time-managed architectures and this wedge issue does not appear to introduce irreconcilable differences between the different architectures. Note that the architecture cannot resolve problems that arise based on the limitations of using simulation systems; the capabilities of systems that cannot respond appropriately to time-managed information will not be improved by changes to the architecture.

2.3.6 Wedge #6: Standard Object Model Provided

DIS	**Yes:** Although not a formally defined object model, the allowable content for data exchange is embedded in the standard. Thus, any application that complies with the DIS standard should be able to interoperate with other DIS-compliant applications.
HLA	**No:** As HLA is a general-purpose architecture, it must be flexible to support a wide range of users. Thus, the design separates the data architecture from the simulation architecture, and simply provides a template to allow users to define their data exchange based on specific requirements. In recognition of the problem associated with specifying a new object model for each new application, users were encouraged to define community-standard object models outside of the architecture. The RPR FOM is

	one example of this.
TENA	**Yes:** To facilitate interoperability among ranges, a standard TENA object model was developed. Thus, if all participants use the TENA object model, a high degree of interoperability can be achieved. However, the TENA architecture also allows users to define alternative object models that better fit their needs. This provides a highly flexible "middle ground" between the options of having the data being part of the simulation architecture or not.
CTIA	**Yes:** To facilitate interoperability, independently developed components and products, the majority of the CTIA Object Model is provided by the architecture. However, some portions of the Object Model are defined as “Flexible Features” which supports extensibility (by individual component developers or product integrators).
ALSP	**Yes**: The simulations in the ALSP confederation are known, and the exercises they support are all similar in nature. Thus, it was possible to achieve incremental agreement on the content of a standard object model that evolved over a period of time and then stabilized.

This is a minor consideration in assessing architecture compatibility and is most closely related to pre-runtime procedures. Gateways are not required simply because of the lack of a common model (although gateways or some other translation mechanism will be required to support communication between applications using different object models). HLA could be modified in this area by officially adopting a standard set of Base Object Models (BOM) that could be flexibly composed to suit multiple needs; this set should include models that would be compatible with the TENA standard model and those developed as parts of the RPR FOM effort, at a minimum. This wedge issue does not appear to introduce irreconcilable differences between the different architectures.

2.3.7 Wedge #7: Object Model Extensible

DIS	**No:** Again, although the DIS data architecture does not include an object model per se, the Protocol Data Unit (PDU) content/structure was defined using an open standards process and resulted in a community consensus to meet identified needs. However, changes to the PDU content/structure currently happen very infrequently, and defining special-purpose experimental PDU’s requires operating outside of the standard.
HLA	**Yes:** Early in HLA development, the static nature of DIS PDU’s was identified as a significant problem. The real world is always changing, and a flexible object model capable of modeling changing data without having to continuously change the underlying standard was designed. The template approach used by HLA (separating data from the simulation architecture) provides the necessary object model extensibility. The Modular FOM feature developed by the HLA Evolved PDG will even provide runtime object model extensibility.
TENA	**Yes:** The need for object model flexibility and extensibility was also apparent during the design of TENA requirements. TENA provides a standard object model, but mechanisms exist within the architecture to extend the object model with whatever new classes or capabilities that are required.
CTIA	**Yes:** Individual component developers or product integrators can extend some portions of the CTIA Object. These features are defined using eXtensible Markup Language (XML) and are parsed at runtime by CTIA Services and components.
ALSP	**Yes**: The content of the ALSP object model is at the discretion of the confederation members. If there is a need to extend the object model, the members can simply agree to adjust their interfaces accordingly.

This is a minor consideration in assessing architecture compatibility and is primarily an issue when connecting to DIS federations (normally, the RPR FOM is used). Gateways are not required simply because of the lack of extensible models in all applications (although gateways or some other translation mechanism will be required to support communication between applications using different object models). A RPR FOM - like object model (for DIS) should also be specified to ensure

compatibility with the other standard object models. This wedge issue does not appear to introduce irreconcilable differences between the different architectures.

2.3.8 Wedge # 8: Data Filtering Supported

DIS	**No:** DIS was designed to achieve interoperability among a limited number of simulations where scalability is not a paramount issue. Thus, using a broadcast protocol was perfectly acceptable, and filtering was simply not required. Modern day DIS users sometimes use subnets to achieve some basic level of filtering, although this is not part of the architecture.
HLA	**Yes:** Being a general-purpose protocol, very large training and experimentation environments were considered part of HLA's user base. For such applications, scalability is critically important, and thus the architecture needed to possess sophisticated filtering mechanisms. The Declaration Management services provide basic publish-subscribe capabilities, while the Data Distribution Management services allow arbitrary filtering regions to be established for any object attribute or interaction.
TENA	**Yes:** Scalability issues also drove TENA developers to implement publish-subscribe capabilities for object state information. Currently, such services are satisfying user needs, and thus there hasn't been a requirement to implement additional capabilities (e.g., region-based filtering such as DDM) in the architecture. A DDM-like capability has been expressed as a TENA requirement and this capability is planned as a very near-term enhancement.
CTIA	**Yes:** The architecture supports “perfect” event and object filtering, at a cost. The cost of choosing perfect filtering is in terms of performance of the (logically) central services. Component developers have the option of utilizing perfect filtering or performing client-side filtering.
ALSP	**Yes**: Filtering for ALSP confederations occurs at both the local level (ALSP Common Module) and global level (ALSP Broadcast Emulator). All filtering is based on publish-subscribe, similar to HLA's Declaration Management services. This provides all the filtering currently needed to support ALSP exercises.

Different requirements for data distribution management (DDM), publish - subscribe filtering (DM), and broadcast approaches can be resolved using gateways. This wedge issue does not appear to introduce irreconcilable differences between the different architectures.

2.3.9 Wedge #9: Reliable Transport Types

DIS	**No:** DIS was designed for real-time applications only. For real-time applications, the latency penalty for reliable transport is generally considered too high a price to pay, as occasional dropped packets in real-time human-in-the-loop environments are not considered much of a problem. Thus, the User Datagram Protocol (UDP, as compared to the reliability provided by Transmission Control Protocol or TCP) was all that was required.
HLA	**Yes:** For many of the constructive users of HLA, repeatability requirements require the use of reliable transport mechanisms. In addition, some federation management and ownership management services require reliable transport. However, since HLA also supports users with low latency requirements, transport can be defined as either best effort or reliable on an individual object attribute and interaction basis.
TENA	**Yes**: TENA can provide both best effort and reliable transport.
CTIA	**Yes**: Nearly all CTIA transport mechanism use some form of reliable transport. Most forms rely on TCP to provide reliability. Non-TCP reliable communications are also specified for point-to-point as well as multicast distribution of data.
ALSP	**Yes**: Reliable transport is all that is used in ALSP exercises. Lost data packets would have a strongly negative effect on the validity of ALSP exercises.

This difference does not seem easily reconcilable. Gateways can be used to ensure reliable delivery as necessary, but can only ensure reliability after that message is received at the gateway. If messages are "lost" before arriving at the gateway, the gateway cannot resolve the problem. The problem can be resolved to a large degree (probably perfect resolution is possible) by interposing a "message-reliability" gateway between each message-generating application and the simulation network (using "middleware", as opposed to the conventional use of gateways between different architectures). This strategy would allow designating messages as reliable transport type before those messages ever appear on the transport network. However, the strategy would likely require many more gateways ("middleware" copies) to be used than the conventional gateway-use strategy and could thus impose performance costs on the simulation exercise as a whole. However, it is not impossible to resolve the differences here and this wedge issue does not appear to introduce irreconcilable differences between the different architectures.

2.3.10 Wedge #10: Support Transfer of Ownership

DIS	**No: In DIS, the simulation that created the entity owns all entity-state information**. The need to transfer this ownership was never a part of the DIS user requirements.
HLA	**Yes:** Some HLA users have expressed the requirement to dynamically transfer attribute ownership during execution. For example, low-fidelity mission-level models may want to transfer ownership of some missile attributes to a higher-fidelity missile fly out model to provide a more accurate representation of endgame. Although these services are not heavily used (compared to most others), HLA can support applications that require such services.
TENA	**No:** In TENA, Stateful Distributed Objects (SDOs) are instantiated by participating applications, who become the owners of that SDO. That application has the exclusive right to publish updates of the values of the SDO's attributes. Transferring this right to another application dynamically has not been necessary for the user community that TENA currently supports. However, the ability to transfer ownership of objects is a planned TENA enhancement.
CTIA	**Yes**: Because CTIA is a client-server architecture, all objects are owned by the central services. This allows virtually any component to change the state of any object in the exercise. However, the architecture allows for components to specify the owner of some objects (particularly for exercise participants) and restrict/transfer ownership of some of the state parameters for participants.
ALSP	**Yes**: Reasons for attribute ownership transfer are essentially the same as for HLA. In ALSP confederations, temporarily transferring ownership allows for higher fidelity representations to be injected at critical points in the execution.

This difference seems readily reconcilable. Each architecture allows dynamic creation and destruction of entities, a process that could (inelegantly, but effectively) substitute for full ownership transfer. However, the difference should not impact interoperability between those architectures that offer support and those that do not; no gateways are required based on this difference. This wedge issue does not appear to introduce irreconcilable differences between the different architectures.

2.3.11 Wedge #11: Data Marshalling Support

DIS	**No:** Data marshalling is not addressed in the DIS specification.
HLA	**No (currently):** Data marshalling is not addressed in the HLA specification. Some commercial RTI vendors provide utilities that offer support, but this is not a requirement. Note that encoding helpers will be included as part of the next revision of the IEEE 1516.1 standard, which will provide some support for data marshalling.
TENA	**Yes:** The TENA middleware provides for marshalling and de-marshalling of data as required.
CTIA	**Yes:** CTIA relies in the Common Data Representation (CDR) to specify marshalling for all objects. The CTIA Frameworks provide utilities for marshalling objects, as do all

	commercial CORBA implementations.
ALSP	**Yes**: The ALSP confederation ran on many different types of computers/operating systems. To avoid marshalling problems, all data was passed as American Standard Code for Information Interchange (ASCII) text.

There are advantages and disadvantages to both approaches. Translation between text and binary or different endian representations can be handled very efficiently at the gateway level, and can be used to achieve interoperability between the architectures. This could also be viewed as one case where business model issues have driven some architectural differences (between HLA and TENA as an example). In HLA, vendors who supply this capability with their products should enjoy a competitive advantage, so there is no need to specify this (and other types of utilities and support tools) in the architecture. This wedge issue does not appear to introduce irreconcilable differences between the different architectures.

2.3.12 Wedge #12: Requirements for Tools / Utilities

DIS	**No:** The DIS standard focused on clearly necessary services to support data exchange in a consistent, effective manner, and left the identification and development of supporting tools to commercial tool vendors.
HLA	**No:** The only software application that is specifically called out in the HLA specifications is the RTI. The HLA program philosophy is that a supporting set of tools is very important, but the tools are closely tied to the process model, which is outside of the core architecture.
TENA	**Yes:** TENA defines a "product line" of supporting tools and utilities to assist users creating and managing logical ranges and for working with the TENA common infrastructure. Since using TENA would be very difficult without these tools and utilities, the product line was made a core component to the overall architecture.
CTIA	**Yes:** CTIA defines a "product line" of supporting tools and utilities to assist users creating and managing exercises and ranges. This product line is the Live Training Transformation (LT2) Family of Training Systems (LT2 FTS).
ALSP	**No**: Certain tools, such as the ALSP Control Terminal (ACT) and the Confederation Management Tool (CMT) are required for ALSP confederations to operate properly.

This is not an issue that impacts interoperability (run-time). This could also be viewed as one case where business model issues have driven architectural considerations. As an example, in HLA, vendors who supply additional capability (e.g., supporting tools and utilities) with their products should enjoy a competitive advantage, so there is no need to specify them along with the architecture. This wedge issue does not appear to introduce irreconcilable differences between the different architectures.

2.3.13 “Wedge” Issue Summary

The different “wedge” issues reflect significant design and implementation choices that have been made within each of the architectures and protocols. However, as described above, none of them introduce irreconcilable incompatibilities that would prevent the integration of the different architectures into a mixed architecture event, although achieving such integration is not without cost (including time, dollars, and capability). Two of the wedge issues, dealing with global event ordering and reliable message transport types, are not easily reconciled and the solution proposed above could have accompanying performance degradations. Some experimentation is required to establish the technical implications and associated costs.

2.4 Projected Future Requirements

The survey instrument, use case analyses, and workshop discussions have been used to posit an estimate of likely future requirements. This list of future requirements has not been the product of a formal requirements generation process, which introduces some limitations. However, while the list presented here cannot be viewed as exhaustive, it is useful to assess the potential of each of the existing architectures and protocols to meet future needs. For each projected future requirement, a table below provides the requirement definition, the importance of the area addressed by the requirement, and a synopsis of the problems that could arise if the requirement remains unsatisfied.

2.4.1 Improved Quality of Service (QoS)

Defined	Quality of Service can provide different priority to different users or data flows, or guarantee a certain level of performance (delivery reliability and timeliness) to a data flow in accordance with requests from the application program
Importance	Repeatability can't be achieved without reliability of messaging
Problems	Dropped packets, Delay, Jitter, Out-of-order delivery, Corrupted / misdirected packets and receiver discovery; Reliability leads to potential latency and network loading

It will probably be a future requirement for all architectures to provide reliable transport and other advanced QoS mechanisms when required by user applications. CTIA appears to have the most robust capability in this area, as that architecture is charged with providing reliable service even when faced with unreliable networks.

2.4.2 Fault Tolerance

Defined	A characteristic of a system allowing it to continue to operate, possibly at a reduced level, rather than failing completely, when some part of the system fails.
Importance	A large and costly simulation event fails when someone in the router room trips over a wall plug …
Problems	Faults can't be exhaustively pre-defined. Major fault categories include simulation system failure or network resource failure.

While much of the fault-tolerance behaviors and capabilities must reside in the applications (the simulation systems themselves), there are some architectural capabilities that could help provide improved fault tolerance. These include providing for application redundancy, automatic resign, polling, providing more robust fault detection and resolution mechanisms, monitoring and notification capabilities. These strategies should be architecture-independent and transferable across architectures.

2.4.3 Information Assurance

Defined	A product or technology that provides important security services (e.g. identification, authentication, confidentiality, integrity, availability, anti-spoofing, …) as an associated feature of its intended operating purpose.
Importance	Imagine the world without any level of information assurance … (banking, national security, commerce, …).
Problems	DoD information technology (IT) system information assurance requirements apply but are not always addressed by simulation systems (DoD Instruction 8500.2 requirements for qualification as a "net-centric" system).

Much of the information assurance requirement is being met today, however inelegantly. Better solutions, more tailored to the exact requirement seem possible for the future. However, there are processes that allow the current architectures and protocols to meet the associated requirements.

2.4.4 Interface to GIG

Defined	The Global Information Grid (GIG) is the globally interconnected, end-to-end set of information capabilities, processes, and personnel for collecting, processing, storing, disseminating, and managing information on demand. The GIG includes all owned and leased communications and computing systems and services, software, system data, security services, and other associated services.
Importance	Interfacing simulations with GIG resources allows real-time data and information to flow into (e.g., timely initialization data) and out of (e.g., Course of Action Analysis (COAA)) the simulations.
Problems	Latency, validity of information, security (all the information assurance issues), asset availability, discovery metadata, … the role of M&S …

To some degree, existing architectures can use gateways to interface with arbitrary networks now (potentially including GIG resources). The role for M&S (and the GIG for that matter) is as yet imprecise, so major immediate advances seem premature. However, as currently understood, the requirement seems achievable.

2.4.5 Load Balancing

Defined	Balancing a workload among multiple resources (host computer, networks, gateways, …)
Importance	Repeatability or could inject faults into the system (fault tolerance improved if can avoid resource saturation).
Problems	Latency, system failures …

Allowing asset redundancy will probably be required to redistribute loads to equivalent resources (network, host, gateway, …). Providing monitors to detect situations requiring dynamic load balancing will also be necessary. These requirements do not appear to be incompatible with existing architectures.

2.4.6 Projected Future Requirements Analysis Summary

The estimated future requirements appear to be supportable with enhancements to the existing architectures. There does not seem to be any requirement that necessitates a complete redesign of the architectures or a new start to build a replacement (more capable) architecture.

2.5 Current State Analysis Summary

The current state includes a wide range of user communities, and different architectures and protocols are used across those communities. No single architecture is dominant and, with the exception of ALSP, the user base of all architectures appears to be increasing (although at different rates). The above review of stated requirements, wedge issues, and likely future requirements shows that the existing architectures have much in common and, more importantly, do not have irreconcilable differences.

There are other important aspects of the current state. Many of the problems that are barriers to interoperability have been described above in the sections stating the problem and discussing the tradeoffs between having a single, universal architecture and multiple, more purpose-specific architectures. Finally, community perceptions about the current state are described in a following section (Assertions). (While these community perceptions presented in the Assertions section clearly describe characteristics of the current state, they are also important in helping to define the best way forward. Thus, the Assertions discussion in this report is placed immediately preceding the Initial Strategy Assessment section.)

In short summary, the current state where multiple architectures and protocols exist and are used simultaneously has both advantages and disadvantages. Chief among the disadvantages is that the integration effort to conduct mixed-architecture events (and attendant costs) is very high. Integration can be achieved, but the process could be greatly improved.

3 Defining the Desired End State

The problem definition and requirements analyses allow a succinct vision statement describing the desired end state that should be achieved by applying the strategy outlined in the LVC architectural roadmap. Future actions should ultimately *provide the simulation communities with a single (either conceptually or physically) architecture that provides established levels of service in each DoD context of use and will allow valid interactions between user systems*. The several implied requirements in this statement require some further development.

3.1 Conceptually or Physically Single Architecture

As argued above, striving to achieve a single physical architecture implementation that can be universally applied is not necessarily the correct outcome for this effort. However, if multiple architecture implementations are to exist in the future, it is desirable to reach a state where the existing architectures are so easily integrated that they can be viewed as a single “architecture of architectures”, conceptually a single resource. Thus, the desired end state is agnostic on the number of different architectures or protocols that exist, but does express a requirement that they are, at a minimum, conceptually a single architecture.

3.2 Established Levels of Service

The goal state should, at a minimum, provide all the necessary services that are currently available. There should be no degradation of performance or capability as compared to the current capability.

3.3 Each DoD Context of Use

The goal state must serve as a universal resource. No existing community of users should be abandoned, regardless of their number.

3.4 Allow Valid Interactions

Any system, simulation or otherwise, that uses the architecture’s service should be assured that their interactions will be valid at the syntactic level. This includes, in the case of a conceptually single architecture, interactions between systems that utilize physically different architectures.

4 Candidate Strategies

The strategy for transitioning from the "as is" state of simulation interoperability for LVC environments to the desired end state described in Section 3 involves many complex technical, business, and cultural factors. While the selection of the optimal near- and long-term strategy for LVC interoperability will require a thorough analysis of these factors, it is important to first identify the set of potential strategies that could serve as the basis for roadmap development. The strategies) described below represent a LVCAR-participant consensus of the possible high-level approaches to addressing LVC interoperability issues from which a roadmap of lower-level actions and activities can be derived. In addition to the strategies themselves, a short discussion of the primary advantages and disadvantages of each (from the architectural perspective) is provided.

4.1 Strategy 1: Maintain the Status Quo

In this strategy, no specific actions are taken to unify the current distributed simulation architectures. This can be thought of as the "natural selection" or “distributed, uncoordinated management” strategy, which recognizes that the various architectures will evolve as needed to meet the future needs of each user base, and that when mixed architecture environments are required, the current (but admittedly ad hoc, inefficient, and decentralized) approach of using gateways and bridges will eventually become good enough to meet future needs. Architectures that do not continue to meet user needs in a cost-appropriate manner will eventually die off with no direct intervention necessary.

Primary advantages:

- There is no disruption to existing architecture users.
- No additional investment needed beyond that which the individual architecture sponsor already provides.

Primary disadvantages:

- Known technical/schedule/cost risks for mixed architecture applications will continue indefinitely.
- Possible duplication of effort during normal evolution of the architectures.
- Further divergence among the architectures is likely.
- Additional architectures may be created that would compete with those now available.

4.2 Strategy 2: Enhance Interoperability of Mixed-Architecture Events

In this Strategy, the focus is to create solutions to improve the interoperation of existing architectures in a mixed-architecture environment. Examples of such solutions include establishing standard agreements (e.g., processes, terminology, object models) that cut across the various architectures and improving the performance, reliability and (re) usability of future gateways and bridges. The individual architectures would evolve to support their native user communities, but oversight would be provided to discourage divergence and duplication of effort. Unlike the approaches that focus on creating an end state that includes only a single architecture, this Strategy assumes that there is benefit in having multiple architectures available for use, that the benefit is worth the dollar cost in maintaining different architectures, and that the interoperability problems caused by the need to mix the different architectures can be resolved.

Primary advantages:

- User community requirements continue to be met based on the normal evolution of the architectures.
- Allows users to choose from a diverse set of architectural capabilities.
- Does not impose a "one size fits all" solution.
- Actively improves interoperability while providing no disruption to existing architecture users.
- Multiple architectures will spur competition between providers and likely lead to more rapid innovation.
- Benefits can be achieved incrementally.

- Oversight Board provides a mechanism to arrest the continued development of new architectures.

Primary disadvantages:

- Cross-architecture integration will still be required for mixed-architecture events.
- Funding will be necessary to maintain potentially overlapping capabilities.

4.3 Strategy 3: Encourage and Facilitate Architecture Convergence

This Strategy is very similar to the preceding strategy, with the exception that policy actions and investment incentives would be added to cause the architectures to converge either into a single architecture or into a set of compatible and interoperable architectures. Thus, while the same roadmap actions would be taken with regard to improving both model and runtime interoperability in the near-mid term, this strategy would include additional actions as necessary to achieve some appropriate level of architecture convergence (including the potential for physical convergence) at a specified future date.

Primary advantages:

- Multiple architectures will compete to be the “convergence target”, fostering competition between providers and likely leading to more rapid innovation.
- Needed architectural changes are phased-in to avoid user community disruptions.
- Benefits can be achieved incrementally.
- Eliminates much of the complexity of mixed architecture environments in the long-term if physical convergence can be achieved.

Primary disadvantages:

- Possible disruption to users, in that the final actions to achieve convergence may be unacceptable to existing architecture users.
- Requires that hard choices be made regarding several technical and business model issues, which may provide disincentives for affected users to transition.
- Uncertainty about the degree of convergence that can be achieved results in a potential for failure and poor ROI

4.4 Strategy 4: Select One of the Existing Architectures

In this strategy, an evaluation of how well existing individual architectures would satisfy all identified requirements (including projected future requirements) would be conducted, and the architecture that represents the "closest fit" to future needs will be chosen as the foundation of a single future architecture for LVC. Actions at that point will focus on adding (hopefully reusing from other architectures) features and capabilities needed by users that do not currently employ the chosen architecture, and institute policy and financial incentives to convince affected users to transition.

Primary advantages:

- The end state is a single architecture, which eliminates the risk and additional resource consumption inherent to mixed architecture environments.
- Reuse of existing architecture capabilities reduces cost relative to a new start.

Primary disadvantages:

- Significant disruption to users of non-select architectures.
- Eliminates the ability for users to consider and choose among multiple architectures for the one that best suits their needs (i.e., "one size fits all" problem).
- Despite policy and other incentives, users may decide to keep using their existing architecture if they feel it is more appropriate for their needs (i.e., this strategy has a relatively high risk of failure).

4.5 Strategy 5: Develop a New Architecture

In this strategy, an entirely new architecture would be developed based on current and future requirements for LVC environments. While reuse of the best ideas and implementations from existing architectures would be encouraged where appropriate, this strategy is intended to be a new start to incorporate emerging technologies and modern design paradigms into the baseline architecture structure rather than retrofitting such ideas into existing architecture(s). Policy and financial incentives will be used to spur adoption of the new architecture.

Primary advantages:

- The end state is a single architecture, which eliminates the risk and additional resource consumption inherent to mixed architecture environments.
- The resulting product should provide the most advanced long-term LVC capability.
- The architecture can be developed from the beginning with an understanding of current problems and future needs (i.e., not retrofitting solutions into an existing framework).

Primary disadvantages:

- Significant disruption to users of all existing architectures.
- The more general an architecture, the less powerful it is for solving specific problems - "one-size-fits-all" architecture may not work well for many users.
- Users may refuse to transition to the new architecture despite establishment of policy and/or incentives (high risk of failure).
- Requires largest up-front investment of all strategies (note: long-term costs may be reduced if some architectures can be replaced / retired).

The advantages and disadvantages of the various strategies described in this section provide some basis for describing those that are best suited to meet the near and long-term needs of the LVC community at large. In addition, general observations regarding how LVC environments are being developed and deployed in support of current applications can be used to evaluate the suitability of each strategy. This information is provided in Section 5 of this report. Collectively, this data is used to provide an assessment of these five strategies in Section 6, to explain the basis for eliminating some strategies before completing more detailed analyses.

5 Assertions

The potential strategies identified in Section 4 define a set of alternatives to transition from the current "as is" state as described in Section 2 to the desired "to be" state as described in Section 3. While all strategies have the same basic long-term goals, they will all differ in terms of their implementation. That is, they all vary in terms of their technical implementation strategy, time to implement, business model approach, required standards, and level of investment. The selection of a preferred strategy (as a prerequisite to defining the LVC Architecture Roadmap) will need to consider all of these factors. Quantitative characterizations of the strategies, by subcomponent, are provided in the main LVCAR report. However, there are also other more qualitative factors that must be considered to produce a completely informed decision. These factors (or assertions) represent practical considerations regarding the application of distributed simulation architectures within the LVC community today, and are considered factual by the communities represented on the LVCAR effort. The list of assertions below (in italics) is also accompanied by a discussion of potential future impact and a supporting rationale.

- *Much can be accomplished with the architectures that are available today and nearly all of the existing architectures are being improved to better serve their communities of use.* The various distributed simulation architectures in use within the DoD today have all been designed to meet the needs of one or more user communities. With few exceptions, these architectures have continued to evolve and mature based on changing user requirements. Although the different architectures have different strategies and procedures for managing this evolution (as well as different funding streams, which in turn affects the pace by which the architectures continue to evolve), members of the various user communities seem to be generally satisfied with the features and capabilities provided or planned to be provided by their architecture of choice. The existence of multiple architectures allows users to select the architecture that best meets their needs and thus provides an incentive for architecture developers and maintainers to competitively keep pace with technology and stay closely engaged with emerging user requirements. Further, at least two factors provide evidence that users are interested in having their native architecture serve their future needs as well: 1) user groups exist to advocate architecture evolution and; 2) standing lists of future architectural enhancements are being implemented in some cases.

- *The Department of Defense has not always taken and is not currently using a consistent, coherent approach to managing LVC environments.* The DoD has sponsored programs to build individual architectures (e.g., HLA, TENA) and has remained engaged in the activities to standardize and improve existing architectures on an individual-architecture basis. However, the current situation does not include a centralized, Department-level effort (or responsible organization) charged with managing or even influencing the coherent, compatible development of the architectures as a set of complementary capabilities. Today, each architecture is left to develop capabilities “on their own”, primarily influenced by the expressed needs of their constituent communities. While this approach is able to respond to stated user-community interests, it suffers from limitations that become most apparent when the various architectures must be integrated to support a mixed-architecture need. There is currently no enterprise-wide incentive for user communities to enhance LVC interoperability or events that include mixed architectures.

- *The number of available architectures has increased since the early 1990’s, at least partially, as a result of inadequate management*. Each of the ALSP, DIS, HLA, TENA, and CTIA architectures has been constructed as a reaction to some perceived shortcoming (actual or otherwise) in the set of available architectures. HLA, in particular, was intended to replace the other architectures existing at the time the HLA project was started. However, none of these architectures has fallen into complete disuse; all still have some community of use. More effective management may have prevented the new starts for TENA and CTIA, given their technological similarity with HLA. Better management may have also led to the complete retirement of DIS and ALSP after HLA became widely available for use. Instead,

experience shows that a new architecture, even though technologically advanced and fully capable compared to its contemporaries, will not necessarily replace those architectures. In each case, we have seen that the creation of a new architecture has simply added one more architecture to the set of those available for use.

- *Mixed architecture environments occur as dictated by needs of the application, not because of any inherent benefit in mixing architectures.* It is well known that mixing multiple distributed simulation architectures within the same M&S environment will generally result in additional cost. In cases other than reuse of previously established simulation intercommunication mechanisms (e.g., gateways and bridges), the attendant problems usually require considerable resources to resolve successfully. Thus, it would seem that developers would strive to avoid having to mix architectures whenever possible. The problem is that the interfaces of the simulation applications that are chosen (assumed to be based on their functional capabilities) to participate in a distributed simulation event are generally designed only for the distributed simulation architecture the simulation application most frequently uses. As examples, real-time platform-level simulations normally rely on a DIS interface, simulations that can run either faster or slower than real-time will normally utilize an HLA interface, and simulations that frequently interact with live range assets will normally use a TENA interface. Providing simulation applications with interfaces to multiple distributed simulation architectures is usually seen as cost prohibitive. Thus, working in mixed architecture environments through gateways or other means of cross-architecture communication is often viewed as a more practical strategy.

- *When mixing architectures is necessary, point solutions to bridging the architectures work in most cases where syntactic interoperability is the main concern, although these kinds of solutions may introduce additional latency and information loss for some applications.* As stated in the previous discussion, ad hoc workarounds that allow simulations with varying distributed simulation architecture interfaces to operate coherently at runtime can be resource intensive and are not always reusable from application to application. As a result, the number of implemented intercommunication mechanisms continues to increase, even though they are often little more than re-invented solutions to very similar, previously solved problems. This inefficiency is one of the several problems that the LVC Architecture Roadmap effort was established to solve. However, it is important to note that in the majority of virtual and constructive uses, the operational requirements of the exercise / experiment itself were at least partially (and in most cases fully) met with the workarounds that were put into place. The primary takeaway from this observation is that there do not appear to be any large capability gaps in today's distributed simulation architectures that prevent important constructive and virtual simulation exercise/experiment objectives from being met. However, reducing the technical, cost, and schedule risks inherent in applying and integrating the capabilities provided by today's architectures by defining and implementing mechanisms for improved cross-architecture interoperability is an area where improvements are necessary, particularly as pertains to the integration of live assets.

- *Mixed architecture approaches may introduce certain limitations on the range of services available to participants within the full simulation environment.* When multiple architectures are used in the same event, the differences in the capabilities they offer are often resolved by allowing participating systems to use only those capabilities that are common to all architectures. As an example, suppose that one group of simulation systems typically federate using HLA and that another group of simulation systems typically interact using the DIS protocol. When these two groups of systems must interact across the HLA – DIS boundary, typically using an HLA – DIS gateway, the full set of capabilities in HLA cannot be used throughout the entire set of simulation applications because not all of the HLA capabilities are available in the DIS protocol. A gateway can be constructed to provide some or all of the HLA-only services to the HLA federates (from the gateway forward), but there is no guarantee that the capabilities can be used on the DIS side of the gateway.

- *Many legacy, and even some new, simulations will not transition to using a different architecture, unless there are compelling incentives to do so.* It is often very difficult to modify legacy systems to use a new architecture. In the past, transitioning a legacy system to a newly available architecture has not always been cost-effective; it has frequently proven much more advantageous to simply provide a gateway linking the legacy system to the new architecture. Recent history shows that a gateway solution to legacy system integration generally provides an acceptable level of performance. There is nothing in view that will alter this observation.

- *GOTS-based and COTS-based business approaches are difficult to reconcile within the scope of a single product.* Today's distributed simulation architectures have different strategies for the business aspects of their operation. Some (like DIS and HLA) use a commercial business model, where a reputable external standards organization maintains the core specifications, and commercial enterprises develop and market tools that support implementations of the specifications. Others (like TENA) use a government business model, where a government oversight group maintains the specifications and the government also sponsors the development of supporting tools and other resources that are freely available to approved users. Each of these approaches has certain desirable and undesirable aspects. Proponents of the current architectures considered these factors along with the needs of their user communities, and made an informed selection of a business model to best meet their perceived needs. Future directions that include merging the architectures, or development of a (replacement) single LVC architecture must also find ways to transition the multiple business model state of the current situation. The problem is that the users of the discarded business model will either: 1) suddenly have to pay for resources that they never had to before (GOTS -> COTS) or; 2) will have to throw away working solutions to adopt alternatives that they may consider to be inferior (COTS-> GOTS). In this latter case, there is also the issue of what happens to the commercial vendors that previously supported the original architecture. The point here is that if there is only one architecture, then there will be significant issues associated with how to choose the one preferred business-model approach (e.g., pure COTS, pure GOTS, enterprise-wide site licenses, or a hybrid approach mixing COTS and GOTS across the architecture's middleware, tools, and utilities.) and how to transition users to the new model. Note that the current situation, where multiple architectures exist simultaneously, allows both COTS and GOTS options to exist at the same time, although this may be accompanied by increased cost to both the user communities and the DoD.

- *Cultural and resource issues will be persistent barriers to convincing existing architecture users to switch to a different architecture.* When organizations decide on a preferred distributed simulation architecture (based on the types of applications they generally participate in), they tend to invest considerable amounts of time and money in establishing the necessary infrastructure to support their architecture choice. As an example, organizations will invest in automated tools to streamline the necessary activities to employ that architecture effectively and efficiently, and invest training dollars to create "experts" in the use of that architecture. These types of investments generally pay off, in that the resulting M&S environments are usually effective in meeting sponsor requirements. Suppose a new, "better" architecture alternative is presented to the manager of an established M&S environment. If the architecture that the facility currently uses is meeting their sponsor's requirements, what is the incentive for that manager to throw away a working solution and adopt something entirely new? From the manager's perspective, all this does is force an expenditure of valuable resources to transition from something known to work well to something that may not work well, thus increasing perceived risk. The point is that the promise of a technically advanced solution will not be an adequate incentive for the larger M&S community to transition to a new LVC architecture. Even provided with additional incentives, transitions may not occur due perceived risk or schedule impact. Also, it is unclear if providing adequate incentives is even feasible since users may insist on direct funding to pay all the transition costs (new tools, new training, help-desk support, etc.). Such

funding may be difficult to obtain given the constrained resources and competing priorities of the DoD. Convincing user communities to consider longer-term return on investment in lieu of short-term cost or cultural issues, combined with convincing management and business cases, will be keys to overcoming the barriers.

- *Architectural choices of how to transfer data between applications (syntactic issues) and application-level choices of how to interpret received and encode transmitted data (semantic issues) both have impacts on interoperability.* Efforts on the LVC Roadmap to date have focused on improving the way LVC assets interact at the syntactic level. However, even if the perfect solution to runtime (i.e., syntactic) interaction could be defined and implemented, true interoperability could not be obtained if representational differences (i.e.. semantic) exist among the various LVC components. As an example, if there was only one fully integrated, highly robust runtime architecture, or if there were "perfect" gateways that could eliminate any barriers to inter-architecture communication, the runtime aspects of achieving interoperability would effectively be addressed. However, perfect receipt and transmission of the data moved through the architecture's communications mechanisms cannot ensure interoperability in its fullest sense. Differences in the way entity behaviors or capabilities (e.g., weapon ranges) are implemented across the distributed architecture, or differences in the various representations of the same battlefield environment (e.g. covered and concealed positions) can easily result in unfair advantages for some systems. An outside observer may not be able to detect associated problems (since the data may have been passed around just fine), but in reality, the inconsistent treatment of shared representations might have been invalidating the whole exercise/experiment. Most of those connected to this study seem to agree that the syntactic, runtime interoperability issues are relatively minor compared to the tougher semantic issues. There is consensus that agreement on architectural issues is a necessary but not sufficient precondition to achieving full interoperability. There is also consensus that resolving some of the tough, semantic issues has considerable value because this is also a necessary condition to solving the interoperability problem.

- *Significant improvements in LVC interoperability can also be achieved via supporting data, tool, and process standards.* In a similar theme to the preceding assertion, there are many actions that can be taken to improve LVC interoperability that have little to do with defining the runtime architecture. Many of these involve the development of standards. As an example, agreements on shared object model representations are good candidates for data standards, as are agreements on environmental data exchange and system/subsystem characteristics for building supporting databases. Tool standards are also important, as identifying requirements for and then providing either COTS or GOTS tools for common design/development functions reduces workload and increases the likelihood of effective data interchange. Common standards for tools can also help resolve some of the system integration issues that impact LVC events. Agreements on process standards also improve the ability for disparate user communities to work together in mixed architecture environments toward common goals.

- *There is a need to recognize and account for longer-term trends (e.g., the widely-held perceptions about the advantages of SOA) in the LVC Roadmap.* Most current distributed simulation architectures (i.e., ALSP, DIS, HLA, and TENA) are designed to be peer-to-peer, not service-oriented. There are several good reasons for this, including the need for peer-to-peer, low-latency interactions and the relative immaturity of supporting technologies (like web services). However, the fact that SOA is currently inappropriate for most existing distributed simulation applications does not mean that some or all of the advantages SOA offers will forever be beyond incorporation into simulation architectures. Supporting technologies will continue to advance, as will the definition and implementation of net-centric services. Although it is hard to predict where SOA will go, and what function M&S will eventually play on the GIG, there is a persistent need to monitor future trends to ensure that the LVC architecture can and will evolve to provide appropriate support for future training exercises and analytic experiments. That is, we expect that there will be a need for simulations to

interact with other systems, such as decision support systems, which have been designed for use with service-oriented architectures.

In summary, the currently available architectures are generally meeting the primary needs of their constituent communities today and are evolving to meet future needs as well. History shows that the number of available architectures tends to increase over time and that once a community of use develops around an architecture, that architecture is very likely to continue to be used. By definition, the inter-architecture communication problem only occurs during mixed-architecture events. While these are currently a small percentage of the number of all DoD simulation exercises, the current emphasis on L, V, C integration is likely to increase the number of mixed architecture events. These mixed-architecture events occur based on the needs of the applications in the event, not because of a desire to mix the advantages of different architectures. Mixed-architecture events can be effectively used, based on point solutions for the intercommunication problem. There is a common belief that having only one architecture would lessen the interoperability problem. However, in addition to the many difficulties in moving from the current situation to one of a single architecture, there is considerable evidence that significant aspects of the interoperability issues (e.g., semantic issues) are apart from the architecture. These issues involve how the shared data is interpreted between participants, not how the data is shared.

6 Initial Strategy Assessment

Sections 4 and 5 introduced many relevant factors that should be considered when determining the path forward for LVC interoperability. A thorough analysis of these factors for all of the strategy options is a rather large undertaking, in that the applicability, prioritization, weighting, and scoring of each relevant factor is context dependent (i.e., dependent on the nature of the strategy itself). Early elimination of any strategies that clearly appear to be "undesirable" before the more thorough analysis is initiated is desirable. The purpose of this section is to describe the basis for early elimination of some strategies.

The initial strategy assessment involved the evaluation of the various qualitative factors and other more practical considerations discussed earlier in this report. This resulted in a set of recommendations that were presented to the LVCAR Expert Team. The Expert Team identified several issues related to the prioritization and weighting of these factors, as well as the need to consider additional factors in the assessment. The discussion and resolution of these issues resulted in a revised set of recommendations, which reflected the consensus position of the Study and Expert Teams. These revised recommendations were presented to the study sponsor, resulting in a decision to eliminate 3 of the 5 initial strategies.

The first, most basic architecture strategy is Strategy 1, "Maintain the Status Quo." The major concern with respect to this strategy is that the current inefficiencies and excessive integration and resource requirements inherent in today's mixed architecture environments are intolerable even in the present, and the natural divergence of existing architectures will continue to degrade the situation. Clearly, this strategy does nothing to improve the current situation, by definition. Thus, there has been a broad consensus within the LVC community that the roadmap must include (at a minimum) some direct, immediate intervention to eliminate this divergence, and to create more robust near-term solutions to known interoperability problems. The status quo offers none of these benefits and is thus not considered to be a viable recommendation as either a near- or a long-term solution. This strategy has been eliminated from further analysis.

Strategy 5, "Develop a New Architecture", suffers from the impracticality of its implementation, at least in the near-term. Most who work in the distributed M&S community have experienced the clear failure of policy mandates and, based on that experience, are convinced the similar tactics are unlikely to work in the future. Further, incentives to users of existing architectures to transition will be ineffective if the users believe that their existing architecture solution is already fully meeting their needs (which most do). Developing a new architecture to replace the existing ones will more likely

result in another architecture being added to the existing set; history indicates that few, if any, architectures will be retired as the new one comes on-line. Thus, the combination of potentially high up-front costs, long development time, and (currently) weak justification for an entirely new LVC architecture effectively eliminates this strategy as a viable near-term solution. Note that this strategy may be justifiable (and even desirable) in the longer-term based on evolving user needs, significant technology advances, and the potential that the community may eventually evolve toward this strategy if other selected strategies are not sufficiently meeting user requirements in the future. At this time, development of a new architecture has been eliminated from further consideration as a possible recommendation.

Strategy 4, "Select One of the Existing Architectures", suffers from many of the same issues as the "Develop a New Architecture" Strategy. That is, users of existing architectures seem to be generally satisfied with them, and significant investments have already been made in supporting infrastructure. The users of architectures that were not the "one selected" would be asked to switch to an architecture that they may have already rejected or otherwise deemed to be cost ineffective. Thus, there would likely be significant resistance to migrating to an externally-designated architecture when that migration would require new investments (e.g., software, personnel training, supporting infrastructure) and may not work as well as their current architecture within their domain. Short of a policy mandate (which has historically shown to be ineffective), an orderly transition is unlikely to happen. Thus, while this strategy could become a natural outgrowth of the "Encourage and Facilitate Architecture Convergence" Strategy, simply choosing an existing architecture and compelling other architecture users to migrate is considered to be impractical, ineffective, and unworkable as a near-term solution. This strategy has been eliminated from further consideration.

Having eliminated these three strategies from further consideration, only Strategy 2 ("Enhance Interoperability of Mixed-Architecture Events") and Strategy 3 ("Encourage and Facilitate Architecture Convergence") remain as viable candidates for further evaluation. Based on analyses to date, these two strategies appear to be the most promising for several reasons. Principally, they lack the impracticality of the other strategies while still providing needed improvements in LVC interoperability. Both are designed to prevent continued architecture divergence, and both provide sufficient flexibility to allow mid-term course corrections if evolving user requirements suggest a deviation. Both Strategy 2 and Strategy 3 could readily branch to either Strategy 4 or Strategy 5, or to each other, as the most desirable long-term course. In addition, both provide for the existence of an oversight body that will have vision over all of the separate groups that currently manage and evolve the existing architectures.

While Strategy 2 and Strategy 3 are very similar in these respects, they also have some fundamental differences. As stated above, Strategy 2 is primarily focused on improving mechanisms that will reduce the difficulties and costs of integrating multiple architectures. Thus, there is a recognition that there is value in allowing multiple, partially redundant architectures to exist. However, as discussed earlier in this report, there are several inherent problems with a multi-architecture state. Strategy 2 recognizes these problems, but rests on the assumptions that: 1) many of the problems can be remedied by architecture-independent actions such as specification of standard, component-based object models, creation of reusable interface capabilities (gateways, bridges, etc.), and the other kinds of actions described above; 2) problems that remain are acceptable, given the benefits of allowing the user communities to determine and act on their own needs without centralized direction; 3) multiple architectures provide the best support for matching the right tool to the right task, and; 4) truly redundant, unnecessary architectures will be retired by market forces as increased use of LVC integrated events focuses user community need evaluations on the value of enterprise-wide standardization. Another unique aspect of Strategy 2 is that, while it requires the existence of an oversight body, that body does not have to be able to exert strong control over the disparate user communities. Oversight in this case is intended to help identify potential opportunities for greater cooperation and resource sharing among the various user communities, and to identify possible actions that could prevent divergence. The oversight body would then make appropriate recommendations to the user communities involved. However, the user communities will make final

determinations of actions based on the recommendations. In effect, priority will be given to the user-community evaluation of needs rather than to the enterprise-wide recommendation.

Strategy 3 differs from Strategy 2 in the method of addressing the multi-architecture problem, the role of the oversight body, and in the primacy of effort. First, Strategy 3 takes the view that the many problems inherent in allowing redundant architectural capability to exist clearly outweigh the associated benefits. The existence of multiple architectures is a problem that must and shall be resolved. Gradual convergence of the architectures is a viable strategy to resolve the problem. As an example, if two architectures are so common in their capabilities that there is little, if any, significant technical difference between them, then those two architectures should be gradually converged into a single architecture. Similarly, if the complete set of technical capabilities offered by one architecture is a subset of the capabilities provided by another, the "smaller" architecture should be gradually converged into the larger one. Thus, Strategy 3 seeks to manage gradual convergence of the entire set of architectures where appropriate. Eventually, the convergence process could result in either a single architecture or a smaller set of compatible, interoperable architectures. Managing convergence in this way requires an oversight body that can influence the evolution of the architectures, using a combination of policy, incentives, and disincentives to shape the actions taken by the user communities that control architecture evolution. Finally, while Strategy 3 includes actions that will reduce the costs and problems that arise when integrating multiple architectures, the primacy of effort is given to achieving convergence of the existing architectures.

Analyses of high-level advantages and disadvantages, and consideration of the qualitative factors based on community observations and derived assertions have not been sufficient to determine which of these two strategies is clearly preferable. At this point, a more detailed description of the strategies is necessary. It is important to note however that even if more detailed analyses lead to the determination that architecture convergence is desirable as a mid- or long-term goal, current LVC interoperability problems require solutions that are much more near-term than can be provided solely by Strategy 3 convergence activities. Thus, the recommended technical approach to achieving defined LVC objectives includes some elements of both Strategy 2 and Strategy 3, in effect producing a blended strategy that includes the most desirable components of both. The most desirable outcome will be capable of addressing near-term needs as well as setting the conditions for long-term improvements in the Department's simulation interoperability position. Some recommended activities are designed to address issues particular to specific architectures, while others will be outside (but supportive) of the architectures. As examples, issues such as how to bridge live RF-based communications into IP-based simulation networks, how to bridge multiple security domains, and how to generate simulated voice and video that is sufficiently realistic to inject into live systems all represent runtime issues that will need to be researched and viable solutions developed if live, virtual, and simulated forces are to interact seamlessly in future environments. Potential solutions to such issues should be viewed as Risk-Reduction Investigations (RRI below) that can be developed and completed as part of an independent research program early in the implementation of the LVC recommended strategy. The feasibility of these solutions, as well as the feasibility of various convergence options, can be tested through focused experimentation and the most promising solutions integrated into the longer-term LVC architecture design.

7 Strategic Agility: Preserving Options is Desirable

User community willingness to exploit the opportunities made available by implementation of the selected strategy will be a key to success in any case. Similarly, architecture proponents must be willing to accept the strategy as a desirable way forward. Consequently, several "human factors" issues can determine the ultimate success or failure of either strategy. The impact of these factors is notoriously difficult to accurately predict, so ensuring that the selected strategy includes the agility necessary to adapt to changing cultural situations is very desirable, if not completely necessary. Further, the technical feasibility of some activities necessary to fully implement either strategy will not be fully assured without the completion of detailed experiments and studies that can compare performance of various implementation options. As an example, Section 2 notes that gateways could be used to functionally overcome many of the problems implied by the Wedge Issues. Section 2 also mentions that there could be associated performance degradations. Thus, some experiments will be necessary to assess the true impact on performance (throughput, latency, etc.) before the use of gateways as a solution can be fully validated. Similarly, some service areas appear to be fully suitable for convergence across the architectures. This cannot be fully assessed until prototype implementations can be subjected to performance studies. As a result, both human factors issues and performance issues introduce uncertainties at the time when a strategy must be selected. The best defense in this situation is to preserve strategic agility. The most desirable strategy will not only be able to react to a changing environment, but will also be able to react in a way that minimizes or precludes wasting resources.

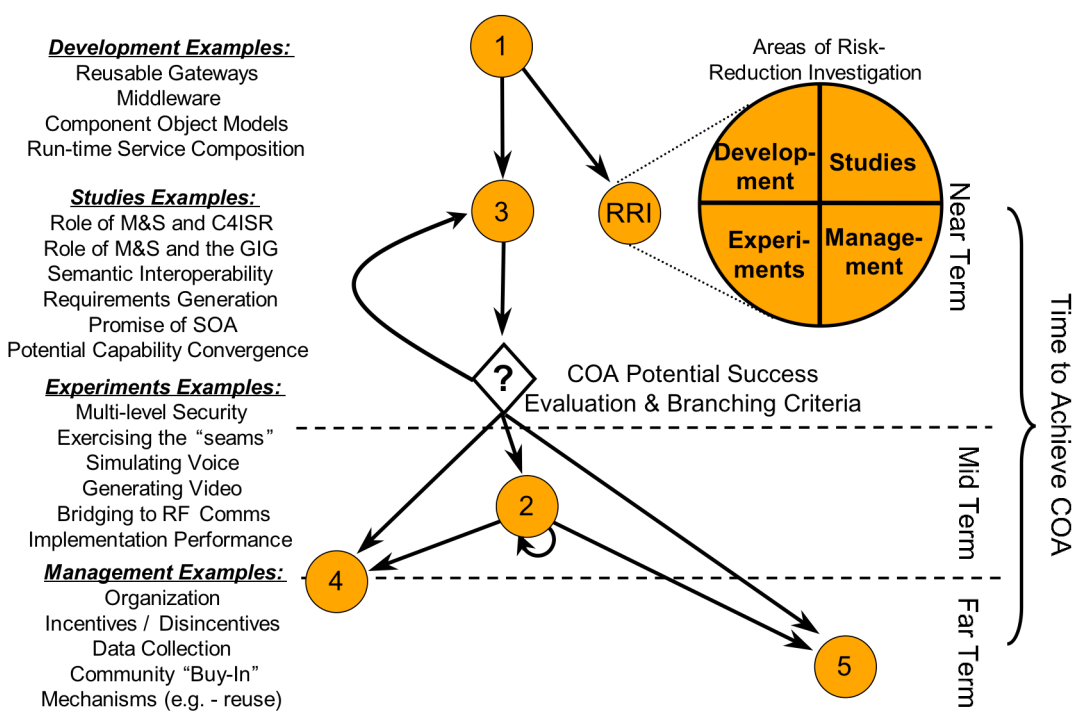

Figure 7.1 Strategy 3: Time, RRI, & Potential Course Correction Options

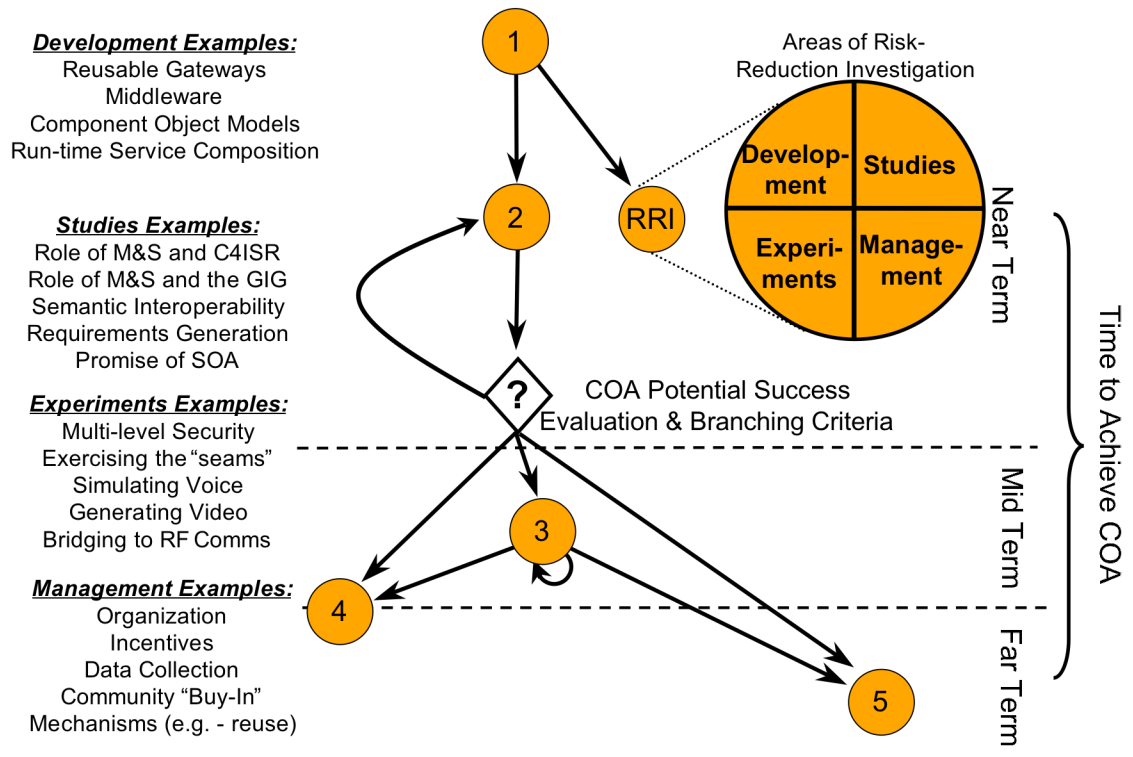

Figure 7.2 Strategy 2: Time, RRI, & Potential Course Correction Options

As briefly described in Section 6, Strategy 2 and Strategy 3 both include sufficient flexibility to make mid-course corrections (branches) if necessary. Figure 7.1 portrays this flexibility as course branches that could occur sometime after a decision point in the far-near-term. Note that the Risk-Reduction Investigations (RRI), as integral components of both Strategy 2 and Strategy 3, should have been completed by this point and some of the results from those investigations may be useful in assessing the utility of possible branches. However, the primary determinants for selecting mid-course corrections will be the success of executing the strategy and the unfolding potential for increased success in other strategies.

As an example, suppose that Strategy 3 ("Encourage and Facilitate Architecture Convergence") is selected and initially implemented as the best way forward, as illustrated in Figure 7.1. The first activities necessary to implement Strategy 3 require completing the activities identified as the RRI and (in parallel) establishing the oversight necessary to achieve convergence. If, after some period of execution, it appears that convergence is not achievable for some reason (e.g., entrenched transition resistance in the user communities, unwillingness of the architecture proponents to implement changes, or the ineffectiveness of centralized direction) and the Strategy 3 execution appears to be failing, it would be possible to make a mid-course correction to adopt Strategy 2 ("Enhance Interoperability of Mixed-Architecture Events") as a compatible strategy that depends less on user willingness to transition, proponent willingness to converge, and the effectiveness of a strong, centralized oversight body. Conversely, if the user communities are accepting transition, but new technologies have become available, it would be possible to branch to Strategy 5 ("Develop A New Architecture") to leverage relevant technology advances. If transition has been accepted and convergence has succeeded to the point that only a single architecture remains as the resource used within DoD, switching to Strategy 4 ("Select One of the Existing Architectures") is a natural result. If some degree of transition and convergence succeeds, but a set of architectures (i.e., more than one) remains in use, then no branching is necessary (the goal state remains as specified in Strategy 3). Very similar branching options are possible if Strategy 2 is executed as the selected way forward, as illustrated in Figure 7.2.

8 General Problems for LVC Events

As indicated in LVCAR survey responses, expert team opinion, and discussions at the LVCAR workshops, many of the problems encountered during the preparation for and conduct of mixed-architecture simulation events are recurrent in nature. While some of the low-level, technical issues that must be resolved to complete specific simulation events can be unique to that event, there is a high degree of similarity in the general nature of the problems between events. The general problems span activities including design, reconciliation, execution & test, and some of the problems can be thought of as more overarching, in that they have an impact in all of these areas. Collectively, they represent the areas that should be addressed by activities designed to enhance the interoperability of systems during mixed architecture events.

The design category includes a set of problems, all of which influence the design of the simulation event and which typically require some resolution before an integration event can be conducted. The problems that could be categorized in this group include:

- Different communities use different systems engineering models and, as a result, when representatives of these different communities must cooperate to produce a mixed-architecture event, differences in process and terminology result in confusion and delay. Essentially, the different systems engineering processes have to be correlated so that the process of designing the event can proceed.
- Because the different architectures and protocols cannot communicate directly, some type of translation resource must be identified or created. Typically, many of the same kinds of resources (gateways, bridges, etc) must be constructed to support each exercise. These resources are usually cost-constrained to be point solutions with little effort expended to improve their potential for reuse.
- Systems that have been built to rely on and use one architecture will not operate on another architecture without non-trivial modifications. The impact is that event designers typically constrain their search for simulation systems that might participate in the event to those that are compatible with a specific architecture, rather than incur the last-resort cost of mixing architectures. In essence, the problem here is that there is no real “plug and play” capability.
- There is some disparity in the services provided by each of the architectures (e.g., HLA provides time-management services while DIS does not). Typically, resolving the service disparity implies that only those services common to all architectures can be used across the entire event - these must be identified and remediation strategies developed, when required.

The problems grouped in the reconciliation category are somewhat more specific (low-level) than those described above and they are more concerned with reconciling differences between groups of simulation systems than with design. In some cases, they could be encountered both prior to and during the integration event itself. The specific problems categorized here include:

- Typically, larger simulation events are designed to connect groups of simulations that may have been used together in smaller events (e.g., a previously designed federation). Each of these previously connected groups of systems will have already reached agreements between the interacting systems on system responsibilities and on the types of objects and interactions that would be allowed (typically included in a Federation Object Model, or FOM, in HLA federations as an example). The same kinds of agreements must be decided for the entire set of systems that will participate in the larger event. Reconciling the previously reached agreements can be difficult because the different architectures use their own mechanisms to express and record the agreements. Essentially, the problem is that federation object models must be reconciled across the different federations that will be brought together.
- The different architectures have either different standard object models or no standard object model. Object models must be reconciled for both syntax and semantics and this is often more difficult than integrating the protocols themselves.

- The different architectures (and different implementations of the same architecture) have made unique, individual decisions concerning the specification of data as it is transmitted among participating systems. As a result, the data encoded using the conventions of one architecture cannot be decoded using the conventions of another architecture. Thus, the data differences must be reconciled and represented in a translator utility that can be interposed between the architectures.

The execute and test problem category includes issues germane to the run-time connection between simulation systems and the testing that must be applied to ensure data is being communicated correctly. These issues include:

- Legacy systems are often included in the larger events and event designers are often very constrained in their ability to modify such systems. Thus these legacy systems usually have to rely on established communications capabilities. Even when legacy system modification is possible, it is usually far more cost-effective to devise a translator than to apply a modification.
- In almost all cases, there is no external testing environment where systems can prepare for the integration event so that almost every test has to wait until the event itself. LVCAR workshop participants who have been responsible for integrating system into such events note that: "There is never enough integration test time with a full up and running federation."
- External systems (e.g., C4ISR) that will be connected to the simulation event "speak their own language" and much like the legacy systems, this language can only be spoken through the use of translators.

Finally, the overarching category of problems include those that span all three of the design, reconciliation, and test & execution areas. These problems include:

- For the most part, there is very little incentive for the different architectures to interoperate. Further, there is no source of available guidance on how they could implement solutions in a more standardized way that would promote interoperability
- Automated tools are not often transferable between architectures as different data formats are involved

9 Creating Resources to Enhance Interoperability in Mixed-Architecture Environments

Strategy 2, "Enhance Interoperability of Mixed-Architecture Events", is founded on the idea that having multiple architectures available for use is desirable and that the best way forward is to take actions that can reduce or eliminate the barriers to interoperability (including the specific problems described above) between the existing architectures and protocols. More specifically, this strategy acknowledges that the existing architectures have been created, have evolved, and are being maintained to meet the specific needs of their constituent communities. Elimination of any architecture should only occur as a natural result of disuse. Modification and management of the existing architectures is left to the owning communities as the best option to ensure meeting the needs of the various user communities, both throughout the DoD and among the Department's coalition partners. To resolve the interoperability problems, efforts should be directed towards creating and providing standard resources, such as common gateways, common componentized object models, and common federation agreements, which can resolve the problems identified in the preceding section and render integration of the multiple architectures an efficient and nearly transparent process. In effect, these actions will create the perception of a single architecture that supports all the diverse simulation systems, even though the systems will actually be serviced by an "architecture of architectures", comprised of as many different architectures and protocols as are required to interconnect the participating simulation systems.

Strategy 3, “Encourage and Facilitate Architecture Convergence”, is clearly related to the first, but focuses on converging services across the architectures as the primary effort. However, the process of convergence will necessarily take place over an extended period of time. Because the LVC interoperability problem is immediate, Strategy 3 also includes the near-term actions that form the basis for the Strategy 2. That is, the same set of actions taken to reduce or eliminate the architecture-integration effort for mixed-architecture events are also included in Strategy 3. Here, they provide: 1) an interim solution during the (possibly prolonged) period of evolutionary convergence and; 2) mechanisms to meet the needs of external systems, which do not use simulation architectures, and legacy systems, which will probably not be revised to take advantage of the converged services, and interface to external systems.

Summing up, Strategy 2 posits that the current state of multiple, somewhat overlapping, architectural capabilities is useful and seeks to take actions that will make these existing architectures more easily integrated, without forcing modifications of the architectures themselves. Strategy 3 recognizes the immediate need for creating the same set of resources to achieve interoperability, but characterizes those resources as interim measures that should eventually become less necessary as the existing architectures converge into a smaller, more compatible and easily integrated set. Thus, the near term requirements for enhancing the current interoperability picture look very similar, regardless of the ultimate strategy.

The near- to mid-term activities that could occur as parts of either strategy cover a wide range. Some are useful only within the context of attempting to achieve architectural convergence. Others involve resources that will reduce the effort required to achieve integration required as part of creating mixed-architecture simulation events. Others are useful in either undertaking. These actions and activities include feasibility studies, planning efforts, implementation efforts, and capabilities analyses. Ultimately, the selected set of activities will constitute the course of action recommended for comprising in the LVCAR Roadmap. The interoperability-enhancing activities that have been identified include:

- Devise the common components of architecture-independent object models
- Produce common gateways and bridges
- Create a common, reusable federation agreement template
- Provide an analysis of the processes and infrastructure supporting M&S asset reuse
- Describe and document a common, architecture-independent systems engineering process
- Produce and / or enable reusable development tools
- Implement a process to maintain specifications for current and future requirements
- Specify a resource to facilitate pre-integration systems readiness
- Determine the feasibility of a common wire-level protocol

The following sections describe each of these potential activities in more detail. Relationships (potential dependencies) between them are also presented. These relationships are used to create a sequence of actions that could be viewed as an initial roadmap of activities that will improve the current state of LVC interoperability. (Note that Section 10 discusses the activities directly related to the convergence required under Strategy 3.)

9.1 Common Components of Architecture-Independent Object Models

Reconciling the differences between the various formats and content of the object models used in different M&S user communities has been recognized as a source of excessive resource consumption when building mixed architecture environments. Providing a common object model, comprised of common components (object model building blocks, sometimes referred to as base object models) and including mappings to current architecture object models, would allow mixed architecture integration to proceed faster and easier.

The purpose of this task is to examine the various object modeling techniques, templates, and methodologies employed by different M&S communities, and determine the best long-term approach for reconciling inherent incompatibilities. There are several possible strategies for achieving this objective. As an example, the rationale behind existing architecture-specific metamodels can be explored, common areas examined, and a standardized template based on a consensus metamodel can be developed. Object model reuse strategies, such as componentization approaches (e.g., Base Object Models) and distribution mechanisms, can also be examined, as well as opportunities for object model content standards and supporting reusable tools.

This activity will directly address the reconciliation problem of differences between object models. It will provide some help in resolving the design-area problems that preclude the applicability of the ”plug and play” metaphor. It will also partially address the overarching problem that there is no guidance on how architectures could evolve in more compatible ways.

The product of this effort would be a set of object modeling standards and resources useful in quickly constructing and implementing object models in a mixed architecture environment. Note that gateways have been developed to translate from one object modeling representation to another. However, such gateways are generally designed only to support very specific translations (e.g., RPRFOM) and represent a source of latency in the simulation system. Eliminating the need for these kinds of gateways by working on the compatibility of the object models themselves is a better long-term solution.

This activity has direct influences on several other activities. The significant influence on the convergence effort is explained above. The existence of a common object model would also greatly simplify the efforts within several other activities. Creating a common wire-level standard and common gateways and bridges will both be simpler undertakings, given the existence of a common object model. Also, the creation of a pre-integration readiness capability will benefit greatly from the existence of a common object model.

9.2 Produce Common Gateways and Bridges

Gateways are currently the most widely used method to link disparate simulations together. Gateways have demonstrated an impressive range of capabilities across the simulation communities that employ them, such as the ability to translate between different protocols or object model representations and to address disparities in the services typically encountered in mixed architecture environments (e.g., time management, filtering, etc.). However, most gateways are designed as point solutions for specific problems, and are rarely shared across user organizations. Thus, the same basic capabilities tend to get developed multiple times, and programs may not even know about more advanced features developed by external organizations.

The purpose of this task is to develop a standard set of gateways, bridges, and automatic gateway generator applications, along with supporting user and developer documentation. A survey of gateway requirements and existing gateway capabilities and technologies will be conducted, and development efforts initiated in areas with capability gaps.

This activity will be helpful in addressing several of the problem areas described above. Common gateways could be used to allow plug and play capabilities. They could also help resolve the “dumbing down” problem by incorporating the “missing” services inside the gateway. They would address the reconciliation area problem by providing the inter-architecture translators as readily available assets. Similarly, they would provide readily available assets to connect both legacy and external systems into the simulation exercise

The product of this effort will be an online repository of "discoverable" simulation gateways with supporting documentation (including V&V information), organized by the application areas and

domains for which the gateways are designed. In addition to the main intent of standardizing a set of reusable gateways (rather than having each program/organization develop their own), there is also a desire to examine emerging software technologies to improve gateway robustness and performance (including extending services offered). Such technologies can be used to address stated deficiencies as part of the overall gateway development plan. As an example, gateways can help to resolve the architecture-dependent aspects of the “dumbing-down” problem for mixed architecture events (however, gateways cannot resolve the individual simulation system limitations that contribute to this problem).

This effort does have relationships to other activities. Creation of a common wire-level standard would reduce the translation burden typically assumed by current gateways. Also, as noted above, the formulation of the convergence plan can help to prioritize the construction of common gateways between pairs of architectures.

9.3 Create a Common, Reusable Federation Agreement Template

Many of the issues that arise when developing distributed simulation environments require the establishment of agreements among all federation participants as a precursor to resolution. While many of these agreements affect interoperability at the syntactic level, others affect the ability to interact with the semantic consistency necessary for environment executions to be considered "valid" (e.g., reference frames, enumerations, data dictionaries). Currently, there is no architecture-independent standard format or content for Federation Agreements documents. Thus, programs must continuously rediscover what types of information require cross-federation agreements, and the lack of a standard format adversely affects the reusability of these products.

The purpose of this task is to develop an architecture-independent template for establishing Federation Agreements, along with potential architecture-specific extensions. Federation Agreements documents developed for existing programs will be examined to define content requirements. In addition to content, the various formats used to capture this information will be examined for applicability to the larger user community. Organized discussions across the community of potential users will be conducted to achieve consensus on content and format.

This activity is primarily designed to address the reconciliation-area problem that arises due to differences between Federation Object Models (FOM). While the activity is not designed to produce a standard FOM, it should produce a standardized template that would permit the FOM reconciliation task to proceed much more easily.

The product of this effort will be an architecture-independent template for establishing Federation Agreements, along with potential architecture-specific extensions. There are no influences between this and any other activity described here. However, any common federation agreement document would have to reference some of the other products (common object models, gateways, and so on).

9.4 Analyze, Plan, and Implement Improvements to the Processes and Infrastructure Supporting M&S Asset Reuse

The existing clearinghouse for M&S information within the DoD is the Modeling and Simulation Information Analysis Center (MSIAC). The Services’ Modeling and Simulation Resource Repository (MSRR) are parts of the MSIAC, thus allowing a wide range of search capabilities relevant to developing or employing M&S applications. However, despite the existence of this on-line service, estimates of utilization indicate that there are relatively few users, and the level of M&S asset reuse appears to be much lower than desired.

The purpose of this task is to examine existing infrastructure capabilities for M&S reuse, compare these capabilities to the required mechanisms for sharing and reuse described in the LVC Roadmap,

and develop and implement a plan to ensure the appropriate discovery and distribution mechanisms are available in the future. The plan for discovery and distribution of reusable M&S assets must address more than just infrastructure. The plan must also include processes for reuse, and address possible incentives to improve reuse within the Department. This implies some analysis of what causes programs to "build new" rather than reuse existing capabilities, and requires that supporting processes and incentives address these tendencies. The eventual product stemming from this effort will be an improved on-line repository that addresses the distribution aspect of all Roadmap products.

This activity is designed to address the design-area problem that intercommunication and other required integration assets (e.g., gateways, bridges) are often built anew each time they are required.

Several other activities depend on the availability of on-line repositories for distribution of its reusable products. However, there are no direct influences that must be considered when conducting either this activity or the others that will depend on the distribution capability.

9.5 Describe and Document a Common, Architecture-Independent Systems Engineering Process

When the user communities of different architectures are brought together to develop a single mixed architecture distributed simulation environment, the differences in the development processes native to each user community represent a persistent barrier to effective collaboration. That is, since these communities must work together toward a common goal, differences in the practices and procedures these communities typically use to build new simulation environments can lead to misunderstandings, misinterpretations, and general confusion. This introduces risk from both the technical and schedule perspectives.

The purpose of this task is to develop a common systems engineering process model for all users of distributed simulation. Existing architecture-specific process models (e.g., IEEE 1516.3, IEEE 1278.3, TENA ConOps) will be examined to identify the key similarities and differences, and a consensus-building process will be instantiated to develop the required products. The product from this activity will be a common systems engineering process model that can be applied across the full range of DoD distributed simulation users. Specific implementation guidance that describes how the common process model should be tailored to meet the needs of specific architecture users will also be included as part of this activity. Finally, an Architecture User Guide will be included as an annex to the core document. This guide would define a mapping between type of simulation event and the architecture or protocol that offers best support for that purpose.

This activity is intended to directly address the design-area problems that stem from use of different systems engineering models. There are no direct influences that must be considered when conducting this activity.

9.6 Produce and Enable Production of Reusable Development Tools

Every development process for distributed simulation includes many opportunities for automation. These opportunities exist throughout every step in the process and include utilities such as requirements development tools, scenario development tools, conceptual and object modeling tools, design tools, networking tools, testing tools, and After Action Review (AAR) tools. In today's environment, there is already a wide range of tools to satisfy each functional need. However, there is also a correspondingly wide range of business models that are used with these tools, including GOTS, COTS, and proprietary solutions. This is a significant impediment to sharing of tools across programs and communities, and increases the cost associated with cross-community collaboration on large mixed-architecture applications. Another key barrier to reuse of tools is that there are many

different formats used by the different architectures to store exercise data. Utilizing a common data storage format would eliminate this barrier.

The purpose of this task is to examine the various business model options associated with efficient, effective sharing of tool resources for DoD simulation applications, identify the most beneficial approach, and implement that approach in a phased, controlled fashion driven by the areas of greatest need. Hybrid approaches will be considered as part of the overall tools strategy. As an example, for some classes of tools, a GOTS solution may be implemented which makes freeware implementations available to all DoD users. For other classes of tools, DoD-wide license agreements may be made with commercial tool vendors to provide the software at no additional cost to approved users. It may also be in the DoD's best interest to encourage a COTS solution for some tool classes, and focus on the development of interface standards so that the commercial tools are interoperable. Concurrently, the different data storage formats used across the various architectures will be examined to determine the feasibility of creating a set of architecture-independent formats for storage of classes of data. Where feasible, a common data storage format will be created.

There will be two products that result from this activity. First, a library of cross-community reusable tools for LVC environment development, along with a business model for tool distribution will be created. Second, a set of architecture-independent data storage formats, specified according to type of data and allowing the tools to operate in different architecture environments, will be described. These products are intended to help reduce overarching-area problems.

There are some influences between this activity and two others. First, establishing a common wire-level protocol would greatly simplify the problem of creating a standard data storage format for some classes of data (such as run-time log data to be used by AAR tools). Secondly, the degree to which common data formats can be designed will impact the technical convergence feasibility activity. In cases where no common data format can be specified, convergence of the associated architectural services appears less feasible. Conversely, if some services can be converged, the convergence will produce a “common data format” as a by-product.

9.7 Implement a Process to Maintain ‘Living’ Specifications of Current and Future Requirements

The first phase of the LVC Architecture Roadmap effort included a “requirements capture” effort, which identified requirements for existing architectures along with qualitative assessments of what capabilities are likely to be needed in future LVC environments. While these requirements provided an accurate snapshot of the capabilities that are supported by today's simulation architectures, any prediction of what will be required in the future is subject to error, due to uncertainties in application priorities and the availability of supporting technologies. Also, this effort resulted in the only specification of its kind; this requirements capture effort resulted in the only comprehensive requirements specification for LVC integrating architectures. In addition to requirements for architecture functional capabilities, there is also a need to capture and maintain, as a living and evolving record, department-wide information detailing the costs and levels of use for the architectures that are currently available. Such information would allow the informed evolution of the architectures, both as individual and collective resources. Thus, the outreach necessary to generate program requirements for LVC environments will also include processes for capturing this additional type of information.

The purpose of this activity is to extend and refine the existing Roadmap development efforts to construct a comprehensive set of current and future LVC requirements. More specifically, this activity focuses on requirements *generation* as well as than requirements capture, and defines and implements a structured process for maintaining a database of evolving LVC requirements based on continuous feedback from LVC users. Examples include requirements that stem from future roles for M&S as a dynamic GIG asset and as a resource for use in C2 operations.

This activity is intended to help resolve the overarching-area problem that architectures do not evolve in compatible ways because there is no available guidance on how such evolution might best proceed.

There will be two primary products from this activity. The first will be a requirements generation process (including the description of a supporting infrastructure that ties in closely with LVC management structures). The second will be a comprehensive specification of DoD-wide requirements, both current and future, for LVC simulation architecture capabilities.

There is no direct linkage between this activity and others in the Roadmap. This activity would run in parallel with other Roadmap activities. Note that the other Roadmap activities may produce requirements that would be captured as part of the requirements database.

9.8 Specify a Resource or Capability to Facilitate Pre-integration Systems Readiness

Simulation system integration into large simulation events can be a very lengthy process of error detection and remediation. The attendant problems would be somewhat reduced if individual or small numbers of simulation systems could be more fully tested for integration readiness prior to integration into the actual exercise event.

The purpose of this activity is to investigate the feasibility of test integration resources, including such capabilities as establishing reusable simulation exercises that include multiple systems using all of the architectures that are currently used in DoD events. Such a resource would necessarily be built to allow join and resign by individual simulation systems over existing networks. The joining systems (e.g., the systems under test) would be required to originate and receive interactions to and from simulation systems residing on a variety of architectures. This activity is intended to help reduce the time required to complete the integration cycle, thus addressing a problem in the execute & test problem group.

The product of this activity will include recommendations on the feasibility of constructing resources or capabilities, as exemplified above, which could be used to assess (and improve) individual systems' readiness for integration into specific events. When a particular capability is found to be feasible, an implementation plan for that capability will also be provided.

The results from other activities will influence the feasibility of constructing the pre-integration readiness resource described here. The existence of common object models, common wire-level standards, and common gateways and bridges will all contribute to the feasibility of this effort.

9.9 Determine the Feasibility of a Common Wire-Level Protocol and a Common API

For those architectures that include an API standard, the wire protocol for intercommunication with other simulations is embodied in the middleware, and thus is completely opaque to most users. However, if the wire protocols used by different middleware applications are different, it is impossible for the middleware applications to interoperate at runtime unless appropriate bridges or gateways are put into place. These translation utilities represent a potential source of error, consume valuable program resources to develop and integrate, and add complexity to the architecture of the M&S environment.

The architectures that include an API could also be studied to determine if a "common API" is feasible. That is, there could be a higher-level API, with its own specification of function calls that could ultimately be translated into the calls defined in either the TENA or HLA API. Use of such an API, in a simulation application, would permit compilation to either of the target architectures (HLA or TENA).

The purpose of this activity is to examine the feasibility of developing a common wire protocol across different architectures and a common (higher-level) API for TENA and HLA. In the case of HLA, it would also involve determining the feasibility of a common wire protocol and API for different versions of the RTI (e.g., RTI-S, HLA 1.3 RTIs and HLA 1516 RTIs). The activity would include an examination of the different wire protocols and API in use today, and identifying and resolving the various technical, business, and standards issues involved with achieving the necessary consensus agreements across the various architecture developers. Note that, in addition to feasibility, this effort should also address some of the desirability aspects of the problem. That is, it may be that both a common wire protocol and API are technically feasible and could be implemented from both a business and standards sense, but it still may not be desirable to pursue either one, either for cultural reasons or for reasons related to the potential impact on commercial developers.

The product of this effort will be a report that either specifies the common wire protocol and or API, or provides the reasons why such products are not achievable. If either a common wire protocol or a common API could be established, it would help resolve many of the problems described in the previous section. These include the lack of a "plug and play" capability, the need for translators between architectures, and the difficulties associated with integrating legacy systems. If implemented, these common resources should be viewed as a resource that ***could be used***, not as one that ***must be used*** in all cases.

As with other efforts that are designed to establish feasibility, this effort should be conducted as early in the Roadmap as possible. The outcome of this effort would impact several of the other candidate Roadmap activities. As already noted above, a common wire-level protocol would facilitate the development of common, reusable tools and utilities, the development of a pre-integration readiness capability. The availability of either product would also simplify the problem of creating common gateways and bridges. Finally, any failure to create either of these resources, particularly the common wire protocol, based on technical infeasibility should also be considered when assessing the technical feasibility of converging the architecture capabilities and services. Also note that a separate activity will be required to implement the protocol, if feasibility is established.

10 Convergence: Activities and Options

Strategy 3, "Encourage and Facilitate Architecture Convergence", is clearly related to Strategy 2, as described above. However, there are also profound differences that set it apart. This strategy rests on the fundamental cost-avoidance concept that the current set of architectures and protocols includes significant overlapping capability and that the differences associated with the implementations of these overlapping capabilities should be reduced whenever possible. Thus, this strategy focuses on convergence, a managed process defined as:

"A process intended to reduce the incompatibilities between competing simulation architecture implementations. Convergence relates to both the types of services offered by different architectures and the way similar services are implemented within the architectures. Convergence may or may not lead to a physical merging of architectures, depending on business considerations and the desires of the affected user communities."

In this strategy, the existing architectures will be converged by modifying those architectures that now provide similar services so that they actually provide the same service. As an example, HLA, DIS, TENA, and CTIA all provide some information filtering capability to serve the common goal of reducing both network loading and the message receipt burden of participating simulation systems. While these architectures and protocols offer similar information filtering services designed to meet similar goals, the exact services provided differ, sometimes resulting in the need to take specific measures to integrate the architectures as resources within a single event. However, if at least some of the architectures and protocols could be modified to provide the same services for information filtering, the attendant integration problems would be largely eliminated. The "information filtering" capability is only a single example of the several areas where convergent modifications to the

different architectures could be applied, all aimed at reducing the meaningful differences between them. In effect, convergence would be applied as an evolutionary process that could eventually result in a smaller set of more compatible architectures. In the limit, if full convergence could be achieved, all of the existing architectures would become so similar that they would effectively become a single architecture.

Altering the ways in which fundamental architectural services are provided to achieve a common implementation across several architectures is a technically demanding undertaking. The difficulty of this task should not be underestimated and a significant amount of up-front analysis and design needs to occur prior to applying modifications to any of the architectures. Modifications should only be applied when they do not change the fundamental nature of the architecture itself and when the modifications can be made in a way consistent with the other services provided by that architecture. As examples, any attempt to add reliable messaging to the DIS protocol would require very fundamental changes to its basic nature and probably decrease its primary value of *simplicity while providing basic functionality*. Similarly, attempts to uniformly provide TENA's Remote Method Invocation (RMI) capability in the HLA architecture conflicts with the partial object orientation of the latter and could thus collide with other HLA services. These kinds of decisions can only be made after prototypes have been subjected to experimentation that can inform a very detailed and technical evaluation and analysis. As a result, two activities (similar to those described in Section 9) are necessary prior to undertaking steps towards full convergence. These are:

- Determine technical feasibility of converging specific services
- Design a plan for convergence

These additional activities, described in the following sections, are intended to perform evaluative analyses of the more problematic areas that could be convergence targets. We also note that there are some areas that are not problematic and where even rudimentary analyses indicate that convergence is a viable option. As an example, the activity related to standard object models (described in Section 9) is an activity that does not require a feasibility estimate, but will help to converge the related architectures. Other potential areas, where the desirability of convergence seems relatively clear exist as well. These are also discussed in sections that follow.

10.1 Activity: Convergence Technical Feasibility Determination

Technical discussions within the LVCAR Study and Expert Teams have resulted in a general consensus that some degree of architecture convergence would benefit future users of LVC environments. However, while the general goal of convergence is considered desirable, there are many questions regarding the technical feasibility of converging architectures that are presently unanswered, as exemplified above. Such questions need to be addressed prior to making significant investments in the implementation of a convergence strategy.

The purpose of this activity is to examine the technical issues related to architecture convergence, including an assessment of existing architectures to identify candidates for convergence, and an assessment of the implementations of the various architecture services to determine exactly how and where to target convergence activities. This latter assessment will require some amount of active experimentation to determine where the "seams" in the architectures are, and how to best address them. The result of this activity will be a report that describes the technical risks associated with architecture convergence, and if and how these risks could be mitigated.

Based on the initial assessment of potential convergence targets, the result of this activity is very likely to indicate that more than a single LVC architecture will remain in use in the long-term. Initial analyses indicate that some of the existing architectures have fundamental differences that do not lend themselves to complete convergence (as exemplified above), in the sense that they can effectively merge into a single architecture. However, for those architectures, even a partial convergence of services can result in improved long-term interoperability, and is still a desirable goal.

The several service areas that have already been identified as potential targets for convergence include: 1) the transport type offered by each architecture; 2) the type of information filtering provided by each architecture; 3) the ownership transfer capabilities provided by each architecture; and 4) the event ordering capability provided by each architecture. The convergence technical feasibility analyses will initially focus on these three potential areas for convergence. In addition to these three areas, several other relevant activities (as described in Section 9) could also serve as general interoperability improvements, regardless of related attempts to converge the architectures. These activities include: 1) creation of common object model components; 2) specification of a wire-level standard; and 3) creating common data formats (included as part of the Creating Reusable Tools and Utilities activity). These three activities are not specifically included as parts of the convergence technical feasibility activity only because they will be examined independently. However, each of them has a direct influence on the degree to which convergence can be achieved.

The convergence technical feasibility analysis is clearly linked to several of the other candidate activities. First and foremost, the result of this effort will determine whether or not any further convergence effort takes place (impacts activities "create a plan to achieve convergence" and "implement the convergence plan"). Also, as described above, the results of the common components of object models, wire-level standard, and common data format (discussed under the topic of Reusable Tools and Utilities) activities influence this activity.

10.2 Activity: Architecture Convergence Plan

The LVC Roadmap will be composed of a time-sequenced series of studies and activities that together implement the core set of objectives identified in the chosen strategy to improve LVC interoperability. However, the Roadmap is primarily focused on near-term activities, by design. The rationale, aside from the obvious unpredictability of LVC requirements and supporting technologies in the longer-term, is that the longer-term LVC architecture evolution strategy depends heavily on the results of Risk Reduction Investigations (RRI). As an example, if the RRI results are positive with respect to convergence, post-RRI activities will mainly focus on the implementation of convergence enablers. However, if the RRI results suggest that the risks associated with an architecture convergence strategy are prohibitive, post-RRI activities are likely to focus on addressing and resolving both technical and cost issues and risks in mixed architecture LVC environments.

The purpose of Architecture Convergence Plan task is to extend and refine the LVC Architecture Roadmap based on the results of the RRI. The RRI results will allow for a more detailed description of the post-RRI path forward, including schedule and cost requirements. The result of this activity will be a revision of the LVC Architecture Roadmap that incorporates the lessons learned from the RRI. This revision will result in additional detail in the post-RRI activity descriptions. However, this activity is not intended to produce a major redirection, but rather a tailoring of mid to long-term activities to account for risk mitigation strategies identified in the RRI tasks.

Development of the architecture convergence plan is dependent on the result of several other activities. Clearly, the technical convergence feasibility will directly influence the conduct of this activity. (Also note that the results of the common object model, wire-level protocol standard, and standard data formats efforts will help to determine technical feasibility of convergence efforts, so all will transitively impact this activity.) There will also be linkages between this activity and the activity that produces common gateways and bridges. When convergence can be planned for two (or more) architectures, then the need for gateways and bridges between those architectures becomes far less pressing. However, if the feasibility estimate or convergence plan eliminates some pairs of architectures from the convergence effort, then the need for bridges and / or gateways between those architectures assumes great importance.

10.3 Initial Assessment of Convergence Targets

The list of capabilities required of each architecture (see Section 2.2) is the logical area to start examining the convergence feasibility question. An initial estimate of the potential convergence that could be achieved in each of these areas is provided below and these estimates should serve as the initial guidance necessary to complete the Convergence Technical Feasibility estimate as described in Section 10.1. Several other factors that are also relevant:

- ALSP no longer has a significant user base and the small base that currently exists is diminishing on its own. In the current situation, influencing convergent change for ALSP is not really necessary or worthwhile.
- We can assume that HLA, DIS, TENA, and CTIA can be influenced to change for convergence purposes:
    - The government still has direct funding influence over TENA and CTIA; barriers here are convincing the architecture owners to effect convergent changes
    - HLA can be influenced through the standards modification process (similar to the recent HLA-evolved 1516 revisions) - revision may not be rapid and there is some risk that desired revision may not be approved
    - DIS changes can be managed through SISO; much as in the HLA case, certainty and pace of desired change are both areas of risk.
- Any changes to effect convergence will also have to account for backwards compatibility.
- Convergence plans must also account for:
    - Multiple versions of the RTI (e.g., RTI 1.3, RTI-S, Different vendor versions, and 1516)
    - Proprietary solutions to the architecture problem – how can these be integrated into federations?
- Federations must also be considered – we also need some focus on enabling "federations to converge", not just on creating converged architectures
- Several of the previously-described activities are also relevant to convergence. These include the activity to create common components of object models, the activity to create a common on-the-wire protocol, and the activity to create standard data formats. Progress in each of this area would also help to converge the associated architectures. (Also note that these areas are not discussed again in the sections below on assessing the feasibility of converging capability requirements.)
- While some improvements can be made (e.g., common object model components, wire-level communication standard, standard data formats, etc.), there is limited opportunity to fully converge DIS with the others (HLA, TENA, CTIA) – the required changes to DIS would change its fundamental nature, making it far less useful for its community of users. Thus, the DIS protocol cannot be subsumed through convergence and it should continue to provide service much as it does today.

10.3.1 Required Capability Area: Publish and Subscribe Information Filtering

The DIS protocol only supports the delivery of "packets" of information, which can be sent as either unicast, multicast, or broadcast messages. Modification of the DIS protocol to include a publish and subscribe capability does not seem feasible; such a change would very likely alter the fundamental nature of DIS and make it much less attractive to those who now use it. Convergence between HLA, TENA, and CTIA appears feasible as all three of these architectures currently provide a dynamic publish and subscribe filtering capability. However, capability can easily be provided using gateways, without (probably) more expensive architectural modifications. There may be some issues with the types of filtering provided (e.g., TENA supports filtering by objects, but not by object attribute which is provided by HLA). In sum, this is an area where convergence between TENA, HLA, and CTIA appears feasible, if not cost effective.

10.3.2 Required Capability Area: Transport Types

The DIS protocol only supports “best effort” delivery of messages. HLA, TENA, and CTIA all provide both best effort and reliable messages. Modification of the DIS protocol to include a reliable message delivery does not seem feasible; such a change would very likely alter the fundamental nature of DIS and make it much less attractive to those who now use it. Convergence between HLA, TENA, and CTIA appears feasible. Some modifications to TENA to allow the simultaneous existence of multiple message types within the same simulation event (i.e., so that a single message delivery type is not the global policy) would be desirable. It is as yet unclear whether or not HLA and TENA need to provide support for unreliable networks (as is required of CTIA) as part of the convergence implementation. If the goal is to create equivalent architectural resources, where one could take the place of another, then HLA and TENA should be enhanced with support for unreliable networks.

10.3.3 Required Capability Area: Interoperate with Other Protocols

The TENA explicit requirement to “interoperate with HLA federations” is typically met using a TENA – HLA gateway. CTIA is required to provide gateways to non-CTIA compliant protocols (such as DIS and HLA). HLA is required to interoperate with other HLA federations. Gateways can be used with each of the existing architectures and protocols, and there is really nothing here that needs to be considered as part of the convergence feasibility analyses.

10.3.4 Required Capability Area: Multiple Message Types

DIS supports multiple types of protocol data units (PDU) as messages. HLA supports interactions and state updates. Both TENA and CTIA support state updates, single messages, and data streams. HLA recognizes the need for transmitting data streams and allows these to be transmitted “outside” of the RTI, although data streams are not included in the standard HLA message types. In this capability area, convergence is probably best supported through modifications to the various object models. Assuming that approach, there is little else that needs to be accomplished.

10.3.5 Required Capability Area: Support Save and Restore Operations

All of the architectures provide (or have a requirement to provide) a save and restore capability. However, this does not appear to be a fruitful area for convergence due the many differences in the applications involved, the limited use of the capability (normally, this type of capability is only used once per exercise, at system initialization), and the likely cost of associated modifications.

10.3.6 Required Capability Area: Support Region-based Information Management (filtering)

DIS has no capability in this area and, again, DIS modifications to provide this type of service seem inappropriate. Both HLA (through Data Distribution Management or DDM) and CTIA provide capabilities to filter information based on geographic position. TENA does not currently support this capability, but has a requirement to provide this enhancement. Aside from DIS, this is an excellent area for HLA, CTIA, and TENA to converge on providing compatible services, particularly since the TENA capability has yet to be implemented. Providing a slightly “relaxed” version of the HLA DDM capability (i.e., do not insist on perfect filtering) would be an appropriate service for HLA, TENA, and CTIA.

10.3.7 Required Capability Area: Support Ownership Transfer

HLA, DIS, TENA, and CTIA all support some capability to transfer ownership from one simulation application to another. There are some differences between their implementations that will require modification to achieve convergence in that HLA allows transfer of object attributes while TENA and CTIA support transfer of complete objects. Both complete objects and object attributes should be transferable in the converged capability.

10.3.8 Required Capability Area: Support Application Synchronization

This capability should be examined as part of the analysis of how best to converge the save and restore capability requirements.

10.3.9 Required Capability Area: Global Event Ordering

DIS, HLA, TENA, and CTIA have all taken different approaches to the time stamping applied to messages (as described in Section 2.2). This does appear to be an area where some level of convergence is achievable and desirable. However, we note that none of the architectures will be able to improve the performance of simulation systems that are not engineered to respond appropriately when presented with out-of-order messages. So, while this is an area where capability convergence is applicable, the root of the interoperability problems lie with using simulations systems, not with the architectures, and a converged capability to a provide consistent global event ordering capability has limited value.

10.4 Summary: Initial Estimate of Feasible Areas for Architecture Convergence

Table 10.1 summarizes the capability areas where convergence appears to be viable. This table includes areas covered in the discussion presented above and in Section 9. The information filtering category in intended to include both publish and subscribe filtering and region-based filtering, as discussed above. Table 10.1 shows that there is some potential for convergence between all 4 architectures but that DIS has the most limited potential for convergence without risking loss of its fundamental advantages.

	DIS	HLA	TENA	CTIA
Transport Types	No	Yes	Yes	Yes
Information Filtering	No	Yes	Yes	Yes
Transfer of Ownership	Yes	Yes	Yes	Yes
Global Event Ordering	No	Yes	Yes	Yes
On-the-wire Standard	Yes	Yes	Yes	Yes
Common Components of Object Models	Yes	Yes	Yes	Yes
Standard Data Formats	Yes	Yes	Yes	Yes

Table 10.1 Initial Estimate of Potential Areas for Capability Convergence

11 Enabling Interoperability Now and Positioning for a Degree of Architecture Convergence in the Future

The activities described in Section 9 will improve interoperability in mixed architecture events as soon as the associated resources can be made available. Some of these activities lead directly to converged architectural capabilities, primarily regarding HLA, TENA, and CTIA. (It is much less desirable to seek a high degree of convergence for DIS because adding the necessary services to that protocol would effectively undermine the advantage it now enjoys; it currently provides the essential intercommunication services without adding excessive user burdens or complexity.) Other potential convergence areas will require evolutionary development, and should only be initiated after a complete program of experimentation and evaluation. If all of the capabilities listed in Table 10.1 are ultimately selected for convergence, they would also serve to improve the mixed architecture integration problem by reducing the potential burdens that would be placed on gateways (e.g., if all architectures could provide the same global event ordering capability, the related burden to be assumed by gateways would be greatly reduced). That is, all of the capabilities indicated in Table

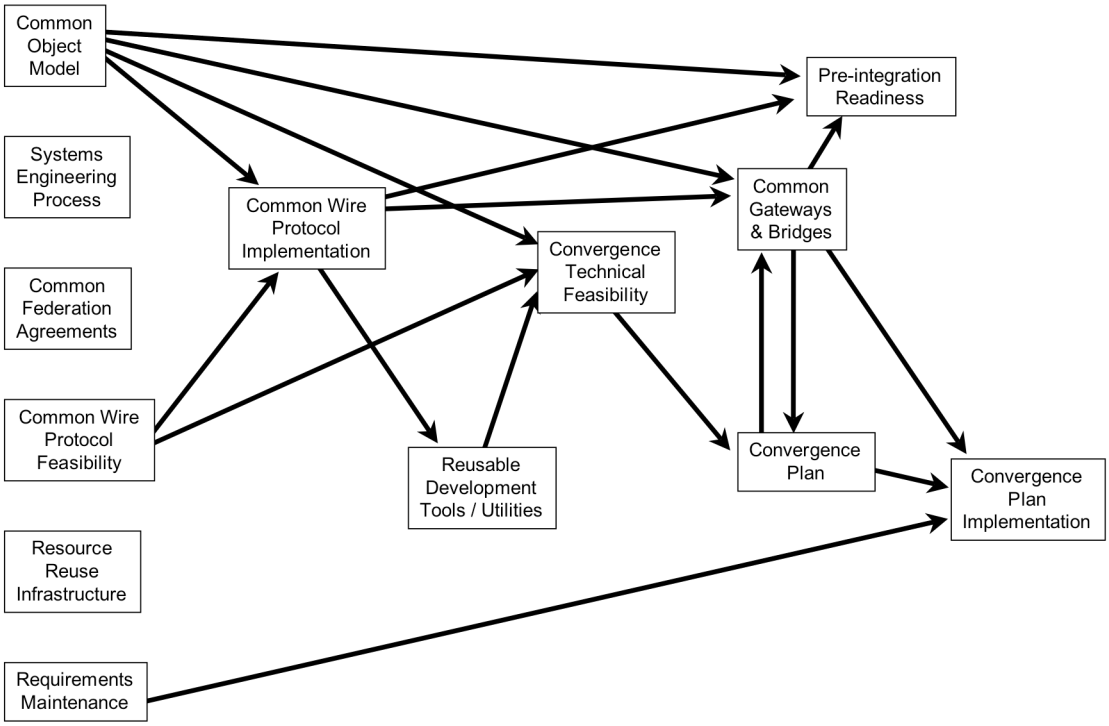

Figure 11.1. Roadmap Activity Relationships and Influences

10.1 could also be thought as individual interoperability-enhancing activities, in addition to those discussed in Section 9. Clearly, there is a somewhat artificial distinction between Strategies 2 and 3 when both are viewed at the technical implementation level. Some interoperability-enhancing activities required as components of Strategy 2 lead to convergence as required by Strategy 3. Some of the likely capabilities that could be converged under Strategy 3 will also enhance interoperability during mixed-architecture events, the focus of Strategy 2.

11.1 Graphical Representation of Activities and Influences

Figure111.1 provides an initial representation of the actions that may eventually be included in the LVCAR Roadmap of recommended activities. In Figure 11.1, actions are spatially arranged to show the influences (as described above) between them; activities (influencing activities) that will have one or more results that will impact another activity point to that activity (the influenced activity). This representation illustrates the temporal relationships between the activities that derive from technical considerations: no influenced activity can conclude prior to the start of all its influencing activities.

Note that Figure 11.1 includes both the interoperability–enhancing activities discussed in Section 9 and the convergence studies discussed in Section 10. Not only are these categories of activities similar at the technical level as described above) but they also have some relationships at the dependency level. Thus, this picture illustrates a blended strategy, where promising parts of both Strategy 2 and Strategy 3 are related, showing those activities that will influence others and the activities that are influenced by others.

Figure 11.1 does not attempt to illustrate the time duration, sequencing, or the relative importance to the community, for each of the activities. A suggested sequencing, where those tasks having the highest potential benefit are sequenced first, is included in figure 11.2. (This arrangement of tasks abides by the dependency relationships illustrated in figure 11.1.)

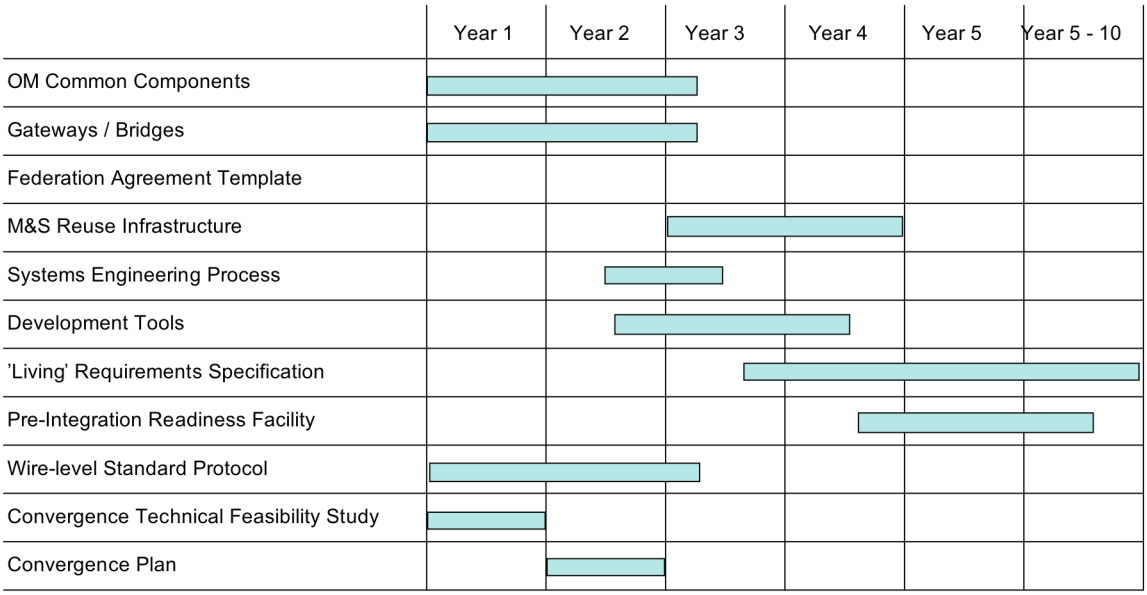

Figure 11.2. Activity Sequencing

11.2 A More Complete Interoperability Problem

The current status of LVC interoperability is fragile and subject to several reoccurring problems that must be resolved (often anew) whenever live, virtual or constructive simulation systems are to be components in a mixed-architecture simulation event. Some of the attendant problems stem from simulation system capability limitations and other system-to-system incompatibilities. Other types of problems arise from the general failure to provide a framework that could help achieve more complete semantic-level interoperability between disparate systems. While these two classes of problems are ubiquitous and often lead to run-time interoperability failures, the LVCAR effort is chartered to address a third category of LVC interoperability problems; these are the interoperability barriers that can be traced to the differences between the architectures and protocols that support the simulation system interactions in mixed-architecture events. While the actions described above can help resolve this third class of incompatibilities, it is important to understand that changes to the architectures will do little to resolve problems in the other two areas. There is little to be done, short of re-engineering, to improve the inconsistent capabilities between simulation systems. A supporting intercommunications architecture cannot change fundamental simulation system behavior. There is however, much that can be done to improve the semantic interoperability situation. While most improvements are not directly tied to architectural services or capabilities, some of the activities described above can have an impact.

During the third LVCAR Workshop, the candidate activities were presented to workshop attendees for their review, comment, and evaluation. The Strategy 2 activity that was evaluated as "most valuable" by LVCAR workshop attendees is also the activity that is closest to solving some parts of the semantic interoperability problem; the Object Model Common Components activity has support across the entire community and there is wide agreement that the benefits promised by this activity

need to become reality as soon as practical. Other topic areas, not included in the sections above, were also presented to the workshop audience for their review, comment and evaluations. These activity areas, while generally assessed as having value by workshop attendees, have not been included in the recommended activities described above because they represent very difficult problem areas that are beyond the scope of the LVCAR effort.

11.2.1 Design Pattern Analysis

"Design patterns" are reusable descriptions of behavior that collectively represent some well-defined pattern of interaction among real world entities. For example, the interaction that occurs among the several systems that together executes the engagement of an air target by a SAM site is a design pattern. Design patterns provide a traceable bridge from system requirements to system design to object models and other simulation environment design/development products. The same reuse strategies that will be examined for object modeling will be also be examined for potential application to design patterns as well.

"Fair Fight" issues, where one simulation within a larger simulation environment has an unfair advantage over another, can be caused by incompatibilities in the underlying assumptions regarding how the real world is being represented. Such problems can be difficult to detect, in that the distributed simulation environment may appear to be functioning quite well. However, incompatibilities in the way the real world is represented internally by interacting federation members can significantly affect the validity of a larger distributed exercise or experiment. The general related need in this area is to define a library of design patterns that collectively represent those aspects of real world operations that are typically reusable from simulation to simulation. This includes an analysis of several different user domains to determine how such operations are currently represented, and what specific domain areas are common across user communities. There are also requirements to focus on the development of methods for exposing incompatibilities in the domain area representations, and also on developing a standardized format and syntax for design pattern representation. Finally, community standards for reusable design pattern content need to be developed.

11.2.2 Algorithmic Consistency

As introduced in the "Design Pattern Analysis" discussion above, incompatibilities in the way real operations are represented in different simulations can significantly affect the validity of simulation interactions. However, even if such representations are completely consistent, "fair fight" issues can sometimes also occur simply due to the underlying algorithms the various simulations use to model real world phenomena. For instance, if two simulations within a distributed simulation environment are representing the exact same radar (with exactly the same system data) but use different algorithms for modeling detection and tracking performance, it is quite possible for different instantiations of the same radar to perform differently simply based on which of the two simulations created the instance. Obviously, such artificialities would not occur in the real world, and thus the validity of the simulation results would be compromised.

The need in this area is to examine the algorithms currently used to model real world systems and phenomena of military interest, and determine if agreements can be achieved across user communities on standards for common algorithms. The scope should be based on an assessment of which algorithms are most common across DoD simulations (e.g., shoot, move, communicate) and where agreements across user communities are most feasible.

In addition to standardization of algorithms, it is desired that implementations of those standards be made available for general reuse in the M&S community. Thus, there is also a need to address the use of repositories for reusable M&S software distribution, as well as the business and cultural issues and associated incentives required to facilitate widespread sharing of models across programs and whole communities.

11.2.3 Environmental Representations

There has been a considerable amount of effort across the Services and various DoD agencies to reduce the amount of effort required to develop and share environmental data for use within DoD simulation events. Examples of success stories include the SEDRIS standard for environmental data interchange and the Environment Scenario Generator (ESG) for production of environmental databases. However, ensuring that use of different simulated environments can result in valid system-to-system interactions remains a persistent problem in distributed simulation events. The notion of “correlating environment representations” permeates the community, without a common understanding of the processes necessary to achieve fully interoperable simulated environments. Thus, there are needs to craft a procedural definition of environmental correlation, identify the main barriers to implementing a complete, consistent environmental representation management strategy, and identify effective implementation policies. Missing resources include a technical strategy and implementation approach for improving the representation of the environment in LVC environments. The implementation approach should address tool requirements and business model considerations.

11.2.4 Human Behavior Modeling

The sheer size of modern battle spaces dictates the need for Semi-Automated Forces (SAF) to simulate human behaviors. That is, there are so many human participants in simulations of current and predicted military engagements that having live operators for all decision-making entities is just not possible. Many live training events use SAF for opposing forces, but many also use SAF for friendly forces they must cooperate with to achieve identified objectives. While there has been considerable research that has gone into this area, and some notable success stories, the requirement for realistic and consistent SAF representations across the full range of entities that play in modern warfare representations is still largely unmet.

The need in this area is to examine the current state of human behavior modeling within the DoD, identify gaps, and develop potential solutions that not only meet gaps in functionality, but also address functional inconsistencies that can occur in mixed architecture environments. The strategy here should emphasize the standardization and sharing of existing capabilities (e.g., JSAF, OneSAF) rather than performing new research. Mechanisms to extract such capabilities from specific M&S systems so that they can be reused in other M&S systems must be emphasized whenever feasible.

11.2.5 Bridging Multiple Security Domains

Issues of security are extremely important when conducting a distributed simulation event. When the entire event is conducted at a single level of security, the mechanisms to ensure that security policies and procedures are being fully addressed are generally well understood. However, many distributed events involve the integration of "enclaves" of simulations that operate at different levels of classification. Current mechanisms to bridge across different security levels are very expensive and time consuming to implement, and verifying that the guards are operating correctly (as part of overall LVC environment testing) is often unacceptably resource intensive.

The need in this area is to examine the issues related to bridging multiple security domains, based on user requirements. Policies, methodologies, and technologies must all be considered in deriving an optimal overall solution for future users of LVC environments.

12 Conclusions and Recommendations

The recommendations presented in this section are based on technical analyses alone. They are provided in support of the integrated study team conclusions and recommendations, which achieve the unification of business, standards, management, affordability and technical area analyses, conclusions and recommendations. The study team integrated conclusions and recommendations

are provided in the overall LVCAR final report. Section 12.2 below provides specific, technical-area recommendations listed in order of decreasing priority.

12.1 Technical Area Conclusions

Each of the existing architectures is providing useful service to a dedicated user community. While there are many similarities between the architectures, there are also differences (largely apart from technical capabilities, but including important cultural and business model factors) so that each architecture has an appropriate role within the entire community. The similarities among the architectures are based on technical service capabilities and these similarities could be exploited and increased to lessen the problems encountered during mixed-architecture integration efforts. One way to enhance the similarities among the architectures is through a process of managed capability convergence, intentionally modifying selected services, by architecture, to create commonality. DIS only presents limited opportunities for convergence. Higher levels of convergence are feasible for HLA, TENA, and CTIA. Apart from convergence, there are also opportunities to improve mixed-architecture interoperability by creating resources (external to any of the architectures) such as gateways, common object model components, and other resources described above. In short, the architectural resources available to the Department are providing a high level of service. Full replacement is not necessary. The need is to make the available architectures capable of easily working together and this need can be met by exploiting opportunities to converge services and by providing interoperability-enhancing resources.

There is also a need to stop the further creation of alternate architectures. There is no evidence that current or future requirements cannot be met by the available resources, either as they stand today or after some level of enhancement. Further, enhancing one or more of the available architectures to meet unresolved needs is preferable to creating “new and improved” alternate architectures, not only from a cost basis, but from an interoperability perspective as well. One lesson of history is that, if a new architecture is created to replace one or more of the existing set, the most likely outcome is that there will simply be one more architecture added to the available set. Actually retiring an architecture has proven very difficult.

There are significant interoperability problems that have little to do with the architectures themselves. Barriers to semantic interoperability are a genuine concern and there is little effort currently directed at resolving the salient problems. Similarly, there are multi-level security issues that impact simulation events.

12.2 Technical Area Recommendations

1. All of the existing architectures should continue to receive support in the immediate future. Subsequent to the conclusion of the convergence service area feasibility experiments, one or more of the architectures may be subsumed by a converged capability, providing a rational basis for ending Department support to the subsumed architecture (s).
2. DIS should only be considered as a candidate for limited convergence. This protocol provides unique services and capabilities that would be lost were the protocol to be fully converged with other architectures that serve different communities and necessarily provide higher levels of service capability. While some of the activities that could lead to more service-level compatibility (e.g., common, components of object models, standard wire-level protocols, etc.) between DIS and the other architectures will prove advantageous, DIS should remain much as it is today, a lightweight, core capability protocol.
3. From a technical point of view, there are significant opportunities to converge the services provided by HLA, TENA, and CTIA. A follow-on effort to conduct additional, more detailed analyses and experiments should be chartered. This effort should be focused on producing qualitative data that can lead to informed decisions on specific service capabilities to be converged. Whether or not such service-level convergence leads to eventual architecture convergence, achieving a state where these architectures provide very similar services will facilitate interoperability and decrease the integration effort when conducting mixed-architecture events.

4. The creation of a common, component-based object model should be aggressively pursued. The Department should leverage existing efforts in this area now underway at JFCOM and ensure that the result spans joint needs across the entire Department.
5. The creation of shared and reusable intercommunication mechanisms (e.g., gateways and bridges) should be aggressively pursued. A follow-on effort to produce the technology required in this area, to include specification of suitable discovery and distribution mechanisms should begin as soon as supporting resources can be made available.
6. Follow-on efforts to complete the analyses and experiments outlined above should be chartered. Again, existing efforts should be leveraged to the extent possible, including the current effort to create a common systems engineering approach that could be used across the Department.
7. A follow-on effort to examine the entire interoperability problem should be started immediately. This effort must address the system-to-system interaction problem at the semantic level and nominate candidate solutions for the “fair-fight” problem, particularly regarding the ways that these problem areas impact the integration of live assets
8. Provisions of the emerging DoD Security requirements must be addressed.