Spring 2003

Creating Enterprise Simulations Using High Level Architecture

Anton R. Lidums
Old Dominion University

CREATING ENTERPRISE SIMULATIONS USING HIGH LEVEL ARCHITECTURE

By

Anton R. Lidums
B.S. May 2001, Old Dominion University

A Thesis Submitted to the Faculty of Old Dominion University in Partial Fulfillment of the Requirement for the Degree of

MASTER OF SCIENCE

ELECTRICAL ENGINEERING

OLD DOMINION UNIVERSITY
May 2003

Approved by:

Roland Mielke (Director)

James F. Leathrum, Jr. (Member)

Frederic McKenzie (Member)

ABSTRACT

CREATING ENTERPRISE SIMULATION USING HIGH LEVEL ARCHITECTURE

Anton R. Lidums
Old Dominion University, May 2003
Director: Dr. Roland R. Mielke

Despite an established need, there are two remaining roadblocks to building distributed simulations and enterprise simulation systems. First, commercial simulation packages and engineering modeling tools are not readily distributable across multiple computers and have traditionally had very little impetus for utilizing a distributable format. Second, although HLA simulations have been built and there are tools to facilitate those efforts, very little has been done or written that specifically describes a "non-standard" integration implementation in sufficient detail to make the process reusable or repeatable. This thesis introduces a proof of concept prototype that was developed to integrate simulations. The paper presents the objectives of the prototyping effort, how various features of the HLA and RTI were used, the design tradeoffs and technical difficulties encountered during development.

Co-Directors of Advisory Committee: Dr. James F. Leathrum Jr.
Dr. Frederic McKenzie

ACKNOWLEDGEMENTS

There are many people who have contributed to the successful completion of this thesis. I extend many thanks to my committee members Dr. Jim Leathrum and Dr. Rick McKenzie for their patience and guidance. I would like to extend a special thanks to my advisor and committee director Dr. Roland Mielke, whose patience, guidance and mentorship have made this endeavor truly meaningful.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AHVPS - Attribute Handle Value Pair Set
ALSP - Aggregated Language Simulation Protocol
API – Application Programmers Interface
ASCII - American Standard Code for Information Interchange
BOS – Battlefield Operation System
C4I - Command, Control, Communication, Computers, and Intelligence
CAE - Computer Aided Engineering
CASE - Computer Aided Software Engineering
CAX - Computer Aided Exercise
CMMS - Conceptual Model of the Mission Space
COBP - Code of Best Practices
COTS - Commercial Off The Shelf
CPU - Central Processing Unit
CSS - Common Syntax and Semantics
DARPA - Defense Advance Research Projects Agency
DHCP - Dynamic Host Configuration Protocol
DIF - Data Interchange Format
DIS - Distributed Interactive Simulation
DLL - Dynamic Link Library
DMSO - Defense Modeling and Simulation Office
DNS - Domain Name Server
DoD - Department of Defense
DS - Data Standardization
DVTE - Deployable Virtual Training Environment
FBE - Fleet Battle Experiment
FED - Federation Execution Data
FEDEP - Federation Execution Development Process
FOM - Federation Object Model
IDE - Integrated Development Environment
IEEE - Institute of Electrical and Electronic Engineers
I/ITSEC - Interservice/Industry Training, Simulation and Education Conference
I/O - Input/Output
IPC - Inter Process Communication
JFCOM – Joint Forces Command
JSAF - Joint Semi-Automated Forces
LAN - Large Area Network
LBTS - Lower Bound Time Stamp
LRC - Local RTI Component
MOM - Management Object Model
MSRR – Modeling and Simulation Resource Repository
NIC - Network Interface Card
NSA - Network Software Architecture
NGNN - Northrop Grumman Newport News

NVE - Network Virtual Environment
OMDT - Object Model Development Tool
OMT - Object Model Template
OO - Object Oriented
OOP - Object Oriented Programming
OOTW - Operation Other Than War
PDU - Protocol Data Unit
RID - RTI initialization Data
RPR – Real-Time Platform Reference
RTI - Run Time Infrastructure
SDREN – Secret Defense Research and Engineering Network
SIMNET - Simulation Network
SISO - Simulation Interoperability and Standards Organization
SOM - Simulation Object Model
SQL – Structured Query Language
STOW - Synthetic Theater of War
TSO - Time Stamped Order
UML - Universal Modeling Language
VASCIC – Virginia Advanced Shipbuilding Integration Center
VBA - Visual Basic for Applications
VDOT - Virginia Department of Transportation
VMASC - Virginia Modeling Analysis and Simulation Center
VNC - Virtual Network Computing
VRML - Virtual Reality Modeling Language
VV&A – Verification, Validation and Accreditation
WAN - Wide Area Network
XML - Extensible Modeling Language

1. INTRODUCTION

This thesis consists of six chapters. The first chapter is an introduction that explains the problem context, problem description, and thesis objectives. The second chapter consists of background for this particular area of research, identification and explanation of HLA rules and processes, and a description of enterprise models. The third chapter contains a high level abstract description for integrating simulations. In the fourth chapter there is a detailed description of the Federation Execution and Development Process for integrating simulations. In the fifth chapter there is a detailed description of the Modify Federate Procedure and methodology for integrating simulations. Finally, chapter six contains high-level conclusions and observations based on thesis work.

1.1 Problem Context

Historians often have referred to significant periods in history as ages; for example, the golden age and the age of enlightenment. The United States is currently going through a transition from the industrial age, where manufacturing was key, to the information age where accessibility, and availability of data, and the ability to process data, are crucial. The paradigm shift associated with this transition introduces many significant challenges for a number of communities. One such community is modeling and simulation.

The field of modeling and simulation (M&S) is broad and diverse. References to modeling and simulation in this thesis are to computer modeling and simulation, and more specifically to the field of M&S, which has been informally defined by academia, industry and the military. The term model can be used to describe anything that is an

abstraction or representation of reality. Models can be as simple as sketches or as complex as high order differential equations and beyond. Reality is complex, and so too is modeling and simulation; as a result there are many methods, tools and processes for representing reality. An important component of M&S is distributed simulation where simulations participate in a networked virtual environment (NVE). NVEs occur via a network software architecture (NSA) that resides on supporting network and computer hardware infrastructure.

As computer architectures have grown in complexity and sophistication, research into distributed computing has produced the capability to conduct simulations using more than one central processing unit (CPU) at a time. Distributed simulation is an extension of this concept. The term is used to describe applications that involve the use of multiple CPUs, usually spread over a local or wide area network. Distributed simulation can be separated into the four communities of interest: distributed simulation in the military community; distributed simulation in the commercial sector; distributed simulation in academia; and the overlap in varying degrees of the previously mentioned sectors. Parallel development efforts in NVE began in the early 1980's; and the DoD has historically been the largest developer of NSA and NVE. Currently the DoD mandates High Level Architecture (HLA) as its NSA. HLA is the product of a number of previously funded DoD NSA efforts such as Simulator Network (SIMNET), Distributed Interactive Simulation (DIS), Aggregated Level Simulation Protocol (ALSP) and Synthetic Theater of War (STOW) [1].

In 1997 the Virginia Modeling, Analysis and Simulation Center was established to promote the commercial use of enterprise simulation and to facilitate the transfer of

simulation technology between DoD and civilian industry. An enterprise simulation is a simulation which is constructed with a top-down view of a business enterprise and which is intended to serve as a decision support tool for decision makers [2]. Enterprise models are often constructed at the entity level. The entities interact within a system representation of the real world or a virtual environment, and potentially a NVE.

There were two main contributing factors that led to the adoption of HLA: first, the work and knowledge from the previous NVE projects; and second, the DoD's greater desire for reuse of legacy products. Advances in hardware, software, and a number of other factors have broadened the range of enterprise simulation [3]. NSA enablers like HLA can extend the utilization and functionality of enterprise simulations by integrating "best of breed" COTS tools, applications, and simulations.

The military simulation community for the most part develops simulations in low level, flexible languages like Java, C or C++, but often desires to incorporate engineering models and simulation applications that do not natively communicate with programming languages. Due to the difficulties associated with working in low-level languages, and the availability of domain specific applications, many commercial organizations and engineers prefer to use more user-friendly simulation tools. Industrial and process engineers may prefer tools such as Arena, ProModel and SLX [4]. Electrical engineers may prefer P-SPICE, MatLab, and others. The list and quantity of tools and domains is vast. While it is relatively simple to use HLA for linking simulations developed using programming languages, alternative methods must be developed in order to create enterprise simulations using HLA with commercial simulation packages and engineering applications. HLA recently was approved as an Institute of Electrical and Electronics

Engineers (IEEE) standard, which has two major implications. First, HLA will become a viable tool for distributing commercial applications, meaning it must be able to be employed relatively quickly by organizations working with limited resources of time, money and personnel. Second, the viability of incorporating commercial simulation packages in military applications must be understood.

Another area of interest is the use of HLA in joint military and commercial applications, particularly the use of HLA by military agencies to incorporate commercial software. These applications can be seen in the case of the Army and Marines using COTS war games and HLA to create virtual battlefields [5]. One example of this is the Deployable Virtual Training Environment (DVTE) that was demonstrated at the Interservice/Industry Training Simulation and Education Conference (I/ITSEC) 2002. The game Operation Flashpoint was used as a training aid for the USMC.

Work is also currently in progress at Northrop Grumman Newport News (NGNN) to integrate war simulations like Joint Semi-Automated Forces (JSAF) with engineering design simulations of an aircraft carrier. NGNN is currently participating in developing a program to model and simulate all aspects of the shipbuilding process, ranging from production of individual components to the operation of ships. In a joint effort, the Navy and NGNN are combing their simulations for mutual benefit. NGNN's models, created in commercial modeling software, send and receive data via HLA into the Navy's War Simulator. This has dual benefit, models in the war simulation are more realistic and the shipyard has the ability to test changes at a low level of abstraction to examine effects on a macroscopic scale. This holistic approach to modeling and simulation allows designers

to evaluate the effects of changes from both the production standpoint as well as the operational and tactical viewpoint [6].

The use of HLA in a purely commercial setting is currently limited, but has strong potential for growth and currently is being researched intensely. Although there are a number of NSAs and NVEs in the commercial sector, there is often significant motivation not to adopt a common architecture such as HLA. The entertainment industry zealously guards its NSAs due to their value. Additionally, industry will often avoid generic NSA implementations in favor of application specific implementations that perform optimally with their particular application. Regardless, there are some examples of HLA use in the integration of Computer-Aided Engineering (CAE) and Computer Aided Software Engineering (CASE) tools. CAD and CASE tools, interfaced using HLA, have allowed several CAD technicians to work individually on portions of a single model in a common workspace [7]. There also are significant efforts to develop web-deliverable, team-distributed training applications [8], medical simulators [9], and logistics tools [4].

1.2 Problem Description

There is a need for distributed M&S. The motivation for using distributed simulation is growing as can be seen through the standardization of HLA in IEEE 1516 [10]. Distributed simulation is an effective means of geographically separating simulations, dividing tasks for effective utilization of processing resources, and generating enterprise models. The Defense Advanced Research Project Administration (DARPA) and the Defense Modeling and Simulation Office (DMSO) have mandated the use of HLA in DoD applications, and have established processes, procedures and support

tools for this purpose. A need for enterprise models also has been demonstrated, but enterprise models incorporate many diverse simulation techniques that are not intrinsically compatible with HLA. For example, in the domain of military battlefield operating systems (BOS), large portions of the simulations are coded in C++. Rather than recoding or coding logistics, troop movement, and supply chain components, these operations could be represented quickly and accurately by commercial simulation packages and then coupled with the military simulation components.

Reusability of previously developed simulations is also relevant to this discussion. Many sophisticated simulations have been developed in academia, government and private industry, a very important development approach is to combine previously generated simulations to model larger and more complex systems. If a repository or library of verified and validated models was established, creating a representation of a complex system would be simplified greatly. Systems could be built by selecting models from a database. The large development time and cost often associated with M&S could be greatly reduced by implementing a black box approach similar to the one used in engineering [11].

Although a need has been established, there remain two roadblocks to building distributed simulations and enterprise simulation systems. First, commercial simulation packages and engineering modeling tools are not readily distributable; tool manufacturers have had very little motivation to develop these tools in a distributable format. Second, although HLA simulations have been built and there are tools to facilitate those efforts, very little has been done or written that specifically describes a "non-standard" integration implementation in sufficient detail to make the process reusable or repeatable.

1.3 Thesis Objectives

A large amount of documentation and literature exists regarding HLA, NVE and NSA. The vast majority of the literature regarding HLA pertains to process for using HLA, theoretical discussions on why HLA is necessary, and descriptions of experiments done with HLA. Very little pertains to the actual implementation of HLA from conception to delivery of a reusable product. The goal of this thesis is to describe "non-standard" integration implementation details for use of HLA in a reusable and repeatable fashion. The approach used to accomplish the stated goal is to present not only the theoretical discussions associated with linking simulations, but also to provide an example implementation. The goal of this thesis is accomplished through the following objectives.

- Provide a sufficient understanding of HLA rules and processes, as well as identification of important HLA references and history, to allow a reader both to create an HLA compliant simulation and to identify the type of simulations that are appropriate for HLA.
- Provide sufficient logistical information for a reader to create a HLA compliant simulation.
- Provide insight into the integration procedure and methodology.
- Provide an approach for creating an application interface with a corresponding Federation Object Model and a strategy for managing time.
- Extend the current use of HLA in battlefield simulations to also include enterprise simulations.

2. MOTIVATION AND BACKGROUND

The arrangement of this chapter is to address the background of HLA and the DoD M&S master plan. The history will provide necessary insight into the next section: Common Technical Framework for M&S. The DoD M&S master plan will establish the link between history and the current need for HLA. This Chapter provides a basic understanding of the Master Plans first objective: "Developing a Common Technical framework for M&S" and enterprise simulation necessary to understand Chapter Four.

2.1 Background

The US DoD was the first organization to recognize the potential of distributed simulation, particularly for interactive applications, and began to develop SIMNET in the mid 1980s [1]. From this program grew DIS and later HLA. Lessons learned and experimentation from the evolution of SIMNET to HLA resulted in the creation of the DoD master plan for M&S and HLA.

SIMNET was the first attempt to address the technical challenges associated with NVE's. The SIMNET testbed linked 11 sites, with 50 to 100 simulations at each site, as an experimental, low-cost, team-training tool for units such as tanks and helicopters. Experimentation in SIMNET resulted in separating the NSA into three basic components: object event architecture; autonomous simulator nodes; and an embedded set of predictive modeling algorithms, also commonly referred to as dead reckoning [1].

In object event architecture, the world is treated as a collection of objects. Objects are items, such as vehicles and weapons systems that can interact; the interactions between objects are called events. Additionally, the basic terrain and structure are separate from objects. The premise, similar to one that exists in object-

oriented programming (OOP), is that the world is treated as objects and events where there exists a logical distribution of computing among the CPU's. Each CPU maintains the state and therefore processing, of its respective objects based on the interactions received.

Predictive modeling algorithms implement an object and ghost paradigm. This paradigm helps alleviate problems caused by packet loss. The principle is that each simulation maintains some minimal representation, or ghost, of relevant objects. Thus, if packets about the ghost object are lost, relatively accurate representations of the entity can be maintained until a proper update is received. For example, a manned flight simulator, federate A, will maintain ghosts or representations of another federate, federate B's, aircraft object instances. Now, if state update information is lost traveling from B to A, A can maintain a sufficient representation of federate B's objects so that loss of information would not degrade an operator's performance in federate A.

An autonomous simulator node means there is no concept of a central server. This has three implications: players can come and go as required; individual failures do not crash the entire simulation; and each node is responsible for one or more objects. As a result of experienced gained from SIMNET, DIS was created.

DIS was an attempt to formally generalize and extend the SIMMET protocol. The goal of DIS is to allow any type of simulation, playing on any type of machine, to participate in a distributed simulation. In addition, DIS is designed to accommodate more simulations than are possible using SIMNET. The first version of the IEEE standard 1278 for DIS appeared in 1993. DIS implements the three-component architecture established by SIMNET in the form of a protocol data unit (PDU). The DIS

IEEE 1278 standard defined 27 different PDU's and is an enormous success [12]. DIS increases the quantity of entities that can be simulated and is the first fully distributed heterogeneous NVE. Any computer that can read or write PDU's can participate. DIS, however, has several limitations. DIS requires large amounts of network bandwidth for large-scale simulations. DIS lacks an efficient method for handling static objects. Static objects are required to send messages at regular intervals since they can be affected by interactions such as explosions [1]. Models and databases must be replicated at each simulator. DIS does not allow users to incorporate new types of information beyond that defined by PDUs. The lessons learned from the transition from SIMNET to DIS resulted in the development of the M&S Master Plan. The purpose of the M&S Master Plans is to help guide progress in the field of NVE and NSA.

2.2 Department of Defense Modeling & Simulation Master Plan

The M&S Master Plan clearly defines and delineates goals and objectives of the DoD. Experimentation in SIMNET revealed a set of core distributed simulation architectural components. DIS formally generalized the NVE but exposed a need for a re-configurable NSA. The creation of these necessary, but limited, point solutions caused DMSO/DARPA to take a step back to create of the DoD M&S master plan. HLA is only one of three sub-objectives of the "Develop a Common Technical Framework Objective". The M&S master plan has six main objectives:

1. Develop a common technical framework for M&S;
2. Provide timely and authoritative representations of the natural environment;
3. Provide authoritative representations of systems;

4. Provide authoritative representation of human behavior;
5. Establish a M&S infrastructure to meet developer and end user needs; and
6. Share the benefit of M&S.

The last five objectives are listed above for completeness and context but are not specifically relevant to accomplishing the thesis objectives [13].

2.3 Common Technical Framework for Modeling & Simulation

"Developing a Common Technical Framework for M&S" is achieved through a set of sub-objectives, which are represented by the following constructs:

1. High Level Architecture (HLA);
2. Conceptual Models of The Mission Space (CMMS); and
3. Data Standardization (DS).

The CMMS and DS are worthy of note because of their influence on HLA and the processes for building HLA compliant simulations [13].

High Level Architecture

The HLA provides the specification for a common technical architecture that facilitates interoperability among simulations and promotes reuse of these simulations and their components. In a HLA application, any number of physically distributed simulation systems can be brought together into a unified simulation environment, known as a federation, to address the needs of new applications. The HLA provides a common framework within which specific system architectures can be defined. HLA does not prescribe a specific implementation, nor does it mandate the use of any particular set of software or programming languages.

The HLA is formally defined by three components. These components are: the HLA Rules that define the key principles of HLA; the Interface Specification; and the Object Model Template, which is a standard means of documenting object models to allow information sharing and reuse of data.

HLA rules are divided into two subsets, the federation rules and federate rules. Federation rules, listed below, ensure the proper interaction of simulations in a federation and identify federate responsibilities.

1. Federations shall have an HLA Federation Object Model (FOM), documented in accordance with the HLA object model template (OMT).
2. In a federation, all representation of objects in the FOM shall be in the federates, not in the run time infrastructure.
3. During a federation execution, all exchange of FOM data among federates shall occur via the runtime infrastructure (RTI).
4. During a federation execution, federates shall interact with the run time infrastructure in accordance with the HLA interface specification.
5. During a federation execution, an attribute of an instance of an object shall be owned by only one federate at any given time.

Federate rules, listed below, ensure federates communicate with the RTI.

6. Federates shall have an HLA simulation object model (SOM), documented in accordance with the HLA object model template.

7. Federates shall be able to update and/or reflect any attributes of objects in their SOM, as well as send or receive SOM object interactions externally as specified in their SOM.
8. Federates shall be able to transfer and/or accept ownership of an attribute dynamically during a federation execution, as specified in their SOM.
9. Federates shall be able to vary the conditions, e.g. thresholds, under which they provide updates of attributes of objects, as specified in their SOM.
10. Federates shall be able to manage local time in a way that will allow them to coordinate data exchange with other members of a federation [14].

The second component of the HLA is the Interface Specification that identifies how federates will interact with the federation and, ultimately, with one another. The specification is divided in to six management areas.

1. Federation Management – This management service handles the tasks of creating federations, joining federates to federations, observing synchronization points, executing saves and restores, and destroying federations.
2. Time Management – The Time Management Service handles the manner federates in a federation respond to the progression of time. There are four options available in the Time Management service,

regulated, constrained, both regulated and constrained, and neither regulated nor constrained.

3. Declaration Management – Declaration Management is the third Management Service. As discussed in the Section 2.1, there is a concept of Object-Event Architecture, which is used to distribute computer processing, based on objects and federates. One of the methods for increasing the number of federates allowed to participate in a federation execution is by reducing the network traffic and bandwidth required. During execution, a particular federates is only capable of processing a limited subset of object instances and, as a result, only information relevant to instantiated objects needs to be received or transmitted. This reduces network traffic. This subset of object instances has led to the concept of "publish and subscribe". Federates that produce object instances must declare what they are able to publish, i.e. generate, and federates that consume object instances must declare subscription interests.
4. Object Management - Object Management is the service that processes registration and instance updates on the object production side and instance discovery and reflection on the consumer side. This service also handles methods associated with sending and receiving interactions controlling instance updates. What this service effectively does is two fold: first, if the federate is one that "produces object instances," it tells federates that "consume them" when an object

instance is created and notifies these federates when something significant occurs to that object instance; and second, if the federate is one that "consumes object instances," it receives the information produced, creates a reflected object, and receives updating information about the state of that object.

5. Ownership Management – The Ownership Management service supports the transfer of ownership for individual object instances. This Management Service also ties back to the three-component architecture concept of an Autonomous Simulator Node. Once an object instance is created, it is the responsibility of the federate that created it to maintain the state and attributes for that object. Ownership management is the service that transfers a federate's object instances to another federate if the managing federate crashes. This service also potentially transfers ownership responsibility if a computer is exceeding its computational capability.
6. Data Distribution Management - This management service supports the efficient routing of data, specifies distribution, and acknowledges routing conditions [14].

The third and final component of the HLA is the Object Model Template (OMT). All objects and interactions managed by a federate, and visible outside the federate, are described according to the standard OMT. The OMT provides a common method for representing object model information. The OMT consists of three main functional components:

1. Federation Object Model – The federation object model introduces all the shared information in a federation such as object and interactions and contemplates inter federates issues such as data encoding.
2. Simulation Object Model – Every federate has a simulation object model. The Simulation object model presents the objects and interactions that can be used external to the federate
3. Management Object Model - The Management Object Model identifies the objects and interactions that are used to manage the federation. MOM objects and interactions are universal to all federates [16].

The goal of HLA, and subsequently the three components, is to facilitate interoperability among simulations and to promote reuse. This, however, is not possible without a CMMS and DS. CMMS provides a context that facilitates the logical distribution of simulation objects and DS creates a common ontology that facilitates reuse.

Conceptual Model of the Mission Space

The CMMS is an abstraction of the real world which serves as a common framework for knowledge acquisition. CMMS, at a very basic level, facilitates developer's decisions about the data to which federates subscribe. As previously discussed, federates use declaration management and object management to publish, subscribe, update and perform a number of other functions on object instances. Developers use CMMS to identify and validate, relevant actions and interactions organized by an entity or organization i.e. red forces vs. blue forces. CMMS provides simulation developers a common representation of the real world and presents actions

and interactions among entities associated with a particular mission area. There is an associated CMMS process available from DMSO, which can be used in designing Federation Object Models (FOM) [13].

Data Standardization

HLA provides a NSA, CMMS provides a context for consistent representation and segregation of data, and DS seeks to facilitate reuse, interoperability, and data sharing among models, simulations and C4I (Command, Control, Communications, Computers, and Intelligence) systems. DS facilitates reuse, interoperability and data sharing by establishing necessary policies, procedures, and methodologies. There are two common products of DS the first, Common Semantics and Syntax (CSS), defines lexicons, dictionaries and taxonomies. An example is a fire interaction. In one federate, this means a weapon is discharged and, in another, this means an entity burst into flames. The second, Data Interchange Formats (DIF), is the physical structure used by programmers to actually interchange data, such as structured query language (SQL), extensible markup language (XML), and others [13].

2.4 Enterprise Simulation

HLA has been used as the key enabler of BOS simulations for evaluation, test and training. However, few software vendors have determined how to take advantage of the benefits offered by HLA with their engineering applications. The DMSO developed a systems engineering process, the Federation Development and Execution Process (FEDEP), as a generalized process for building HLA federations [17]. DMSO created the FEDEP process with the intention of extending HLA to non-BOS simulations,

operations other than war (OOTW), and engineering applications. This demonstrates a natural symbiosis with efforts to create enterprise simulations.

Enterprise simulations in the general sense are dynamic models or simulations constructed with a top-down perspective. Its intended use is to provide an overall conceptual perspective of an enterprise [18]. "An enterprise could be a factory, a port facility, transportation network, urban renewal project, entertainment facility, school, military unit or command." The design characteristics of an enterprise simulation should be created such that an individual who possesses related domain knowledge will intuitively understand the model's outputs and user interface. Differing domains are conducive to differing modeling techniques, strategies, and implementations, which leads to the need for enterprise simulation [2].

Enterprise models necessitate the utilization of various simulation techniques. These models can contain modules best represented as continuous time systems, discrete time systems, or deterministic optimization models [3]. The goal of this research is to demonstrate that HLA can form the underlying structure of an enterprise model for commercial or military use. Enterprise modeling is desirable because it permits the integration of differing simulation packages, allowing model designers to choose simulation tools based on merit, thereby facilitating the most efficient and effective packages to be integrated into the larger system.

3. SIMULATION INTEGRATION

HLA is now a mandatory standard for military simulations and represents the state of the art in distributed simulation. Although HLA is still mainly associated with large-scale, military, distributed simulation projects, the use of HLA is now migrating to civilian applications in the form of enterprise simulations. An area in which HLA applications are of value to the civil sector is in generating solutions for transportation and logistics problems. One of the specific reasons DMSO developed the FEDEP was to extend the applicability of HLA to engineering applications for use in BOS simulation [17]. This chapter uses the background information presented in Chapter Two to investigate the critical issues that, when understood and properly applied, facilitate successful simulation integration and execution of the FEDEP. This chapter consists of three sections. Section 3.1 identifies the challenges associated with using HLA. Section 3.2 identifies a general approach to simulation integration, and Section 3.3 presents a detailed description of hurdles introduced by time and time management, data standardization and data transfer, and FOM creation.

3.1 Challenges of Integration

Integration of a simulation is used to describe the process of combining simulations developed using stand-alone tools into larger, more complex enterprise simulations. Simulation integration can be relatively straightforward when developers have access to source code, the simulations manipulate time in a consistent fashion, and they are coded using an OO paradigm. The reason these conditions make integration straightforward is related to the motivations for establishing the three basic NSA components established by the SIMNET program, and the data standardization issues

addressed by HLA. Engineering simulations, unlike many BOS simulations, often exist in applications that were created with very little consideration of interoperability. These legacy simulations and simulation applications often require both middleware creation and simulation modification in order to utilize HLA. The challenges of building distributed computer systems with simulations and applications that are not inherently open, OO based, time compatible and data compatible are expressed by the following three questions.

1. When and for what type of distributed simulation is HLA appropriate?
2. Assuming legacy models exist that reflect the conceptual model required, what are the evaluation criteria for attempting to either modify the legacy model/application or start with a new model/application?
3. Once a decision has been made to use a particular model/application what process and approach is used for implementation?

Application development using the FEDEP is typically a concise and regimented process when developers can access source code, the source code is written in an OOP language, and an external method exists for manipulation of simulation time. When integrating applications where the previous conditions do not apply, integration becomes a balance and an iterative process. Integration is a balance because the data representation and communication requirements necessitated by HLA and the capacity of the target application to fulfill those requirements are usually in opposition. The target application's architecture may impose limitations on the type, size, and frequency with which data are exchanged. The target application's architecture also has a significant impact on the utilization of external mechanisms to control simulation time, which in turn can impact

the FOM, time management service, and data transfer mechanisms. Some implementation decisions are dictated out of necessity, but often implementation alternatives are available to federate software developers. *A priori* knowledge of integration issues combined with a general approach and appropriate reference material, greatly reduce the time required to alter federates for integration.

3.2 An Approach to Simulation Integration

One of the goals of this research is to determine the feasibility of developing an enterprise simulation constructed using commercially available discrete event simulation packages. Commercial simulation tools are efficient and effective for modeling a large range of industrial processes. As previously mentioned, there are significant benefits to using COTS products. Distribution of development costs, selection of products best suited for an application, and broader user communities comprise several of the benefits. The ability to federate "best of breed" applications and the reuse of legacy simulations can greatly extend simulation fidelity in conjunction with a reduction in development time and cost. A basic approach to creating an enterprise simulation is depicted in Figure 3. Once a requirement for composition is identified, a general systems engineering approach is taken. Requirements are produced to explicitly establish acceptability criteria. A solution set is defined, and then an evaluation is performed to match the applicability of the solution to the requirements. The blocks in blue in Figure 1 indicate the approach taken for this research. Also worthy of note is the bright green decision block. In circumstances where HLA is mandated and the FEDEP is used, the appropriateness criteria may be similar to the evaluation criteria for attempting to either modify the legacy model/application or start with a new model/application [17].

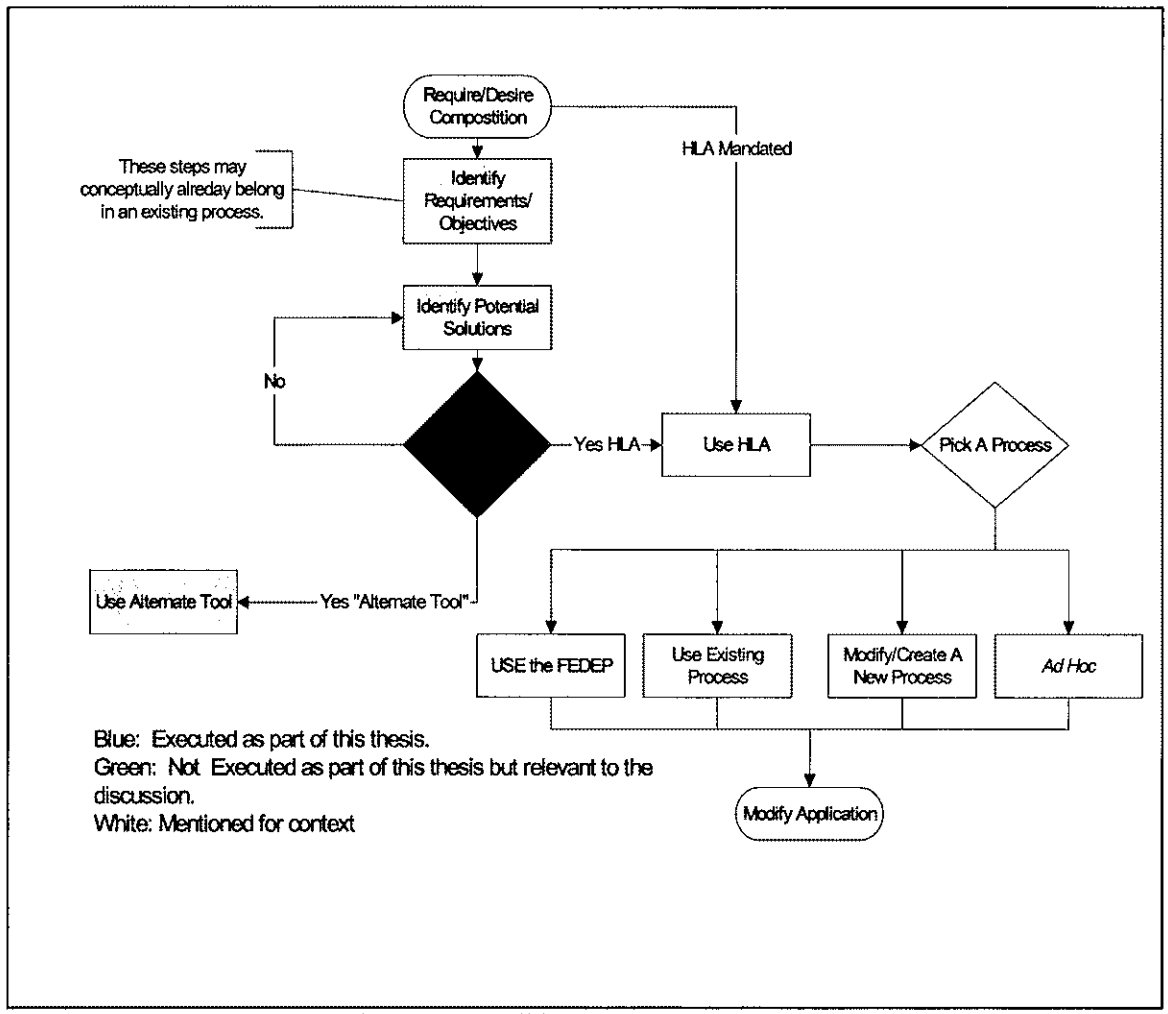

Figure 1. Modify Application Process

HLA was developed for man in the loop computer aided exercise training. The evolution of SIMNET to HLA resulted in object-event architectures, a notion of autonomous simulator nodes and dead reckoning. The implications of this architecture are that distributed simulation systems are sufficiently coupled, as depicted in Figure 2, that events and response events appear realistic to human perception, and sufficiently decoupled such that available CPU cycles and network infrastructure are adequate to support the desired number of object instances.

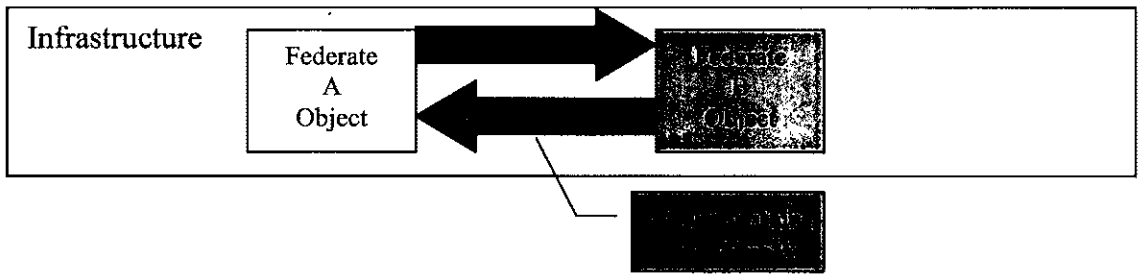

Figure 2. Time Associated with Event and Event Response

The supposition is that HLA is beneficial for time dependent systems with feedback and that the complexity of implementation increases as a factor of the limitations imposed by hardware, the quantity of object instances, and the frequency and quantity of data passed. HLA can introduce a significant amount of overhead, both computationally and monetarily. Systems that do not have parallel processing requirements, feedback, or that necessitate strict adherence to time requirements may be more suited to distributed modeling tools such as Phoenix Integrations' Model Center® or Technosofts' TIE®.

3.3 An Approach to Federate Modification

Section 3.2 describes a general approach for integrating simulations as depicted in Figure 1. This general approach results in selecting a process to modify an application for HLA integration. Figure 3 depicts a continuation of the process established in this research to achieve interoperability. The process continues from the modify application block in Figure 1.

The first major step of this process is to evaluate the accessibility of the target application and select a strategy based on the accessibility. Previous work in the field of

distributed simulation using HLA has introduced a number of strategies for achieving interoperability.

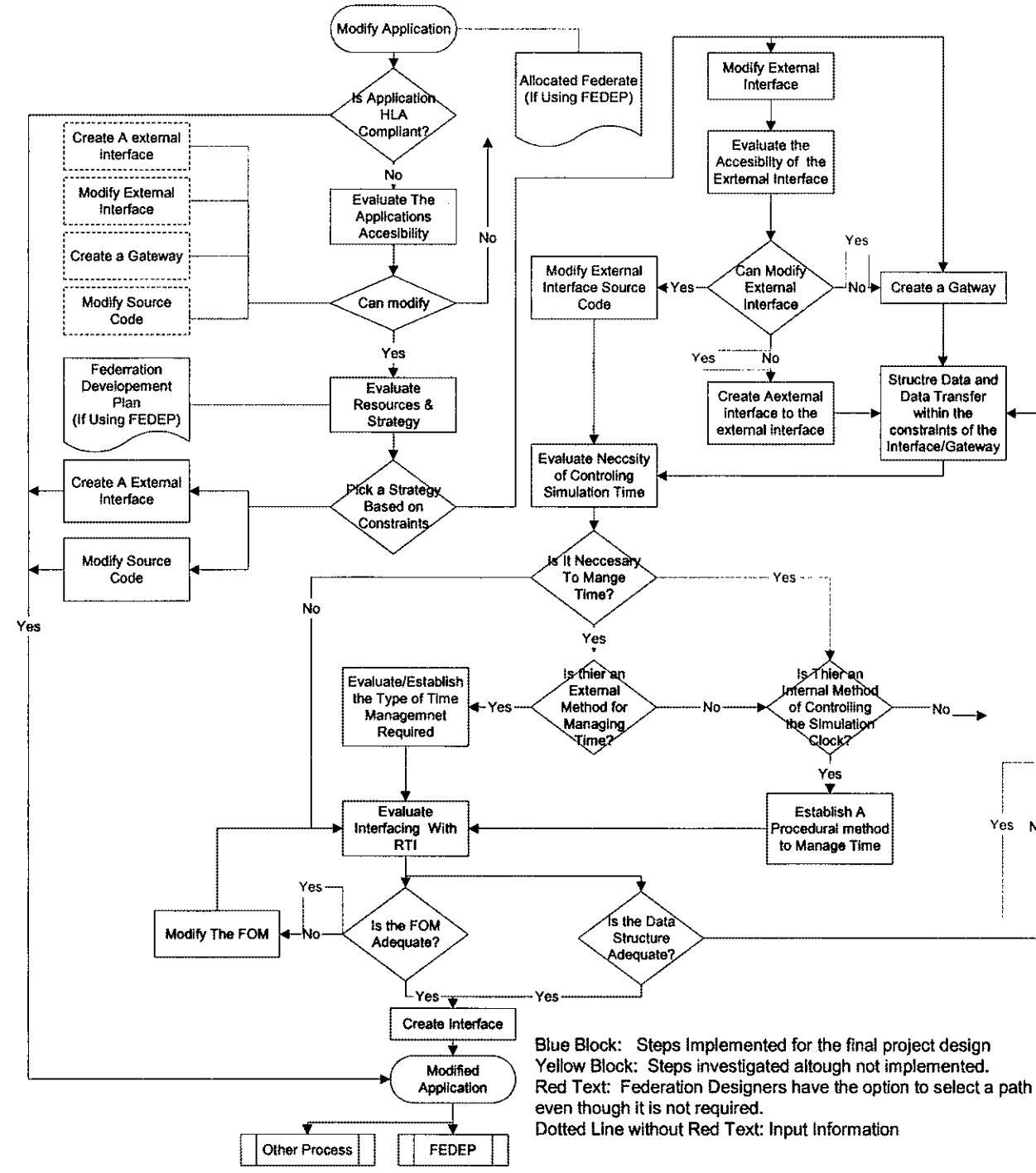

Figure 3. Modify Federate Process

Theses strategies are broken down into two perspectives: the programmer's point of view, and the modeler's point of view [4]. From the programmers perspective, there are a number of alternatives when attempting to interface simulation tools.

Strategy

Programmers can re-implement the application. Re-implementation requires the modification of application source code to be HLA compliant. This approach has a number of considerable benefits as well as several disadvantages. The first advantage is that application changes are potentially transparent to the modelers. The second advantage is there is no overhead introduced by API's and middleware. The third advantage is programmers can format output data as desired; access to source code likely allows precise control over simulation time and data can transported as desired. The first disadvantage is the requirement to have access to source code, which is often proprietary. The second disadvantage is programmers would have to be knowledgeable in both the applications code as well as one of the languages in which the RTI API interface specifications are written. The third disadvantage is there is a strong likelihood large amounts of time would be required to learn and modify the application code.

Another strategy is to extend intermediate code. Some simulations have intermediate code that can be modified, thus making the simulation HLA compliant. The advantages and disadvantages of this strategy are similar to those for re-implementation. The major advantage difference is if intermediate code does exist, it is more likely accessible and would require significantly less time to modify. The disadvantage is this strategy likely provides less flexibility for control of the simulation clock.

The third strategy to achieve interoperability from a programmer's perspective is to create and use an external programming interface. This strategy is useful for tools with open extensible architecture. The general procedure for this strategy is to create a Dynamic Link Library (DLL) interface. A DLL is software that can be called by other programs to perform computations or functions. DLLs can allow data to be shared by multiple applications via shared memory [20].

Using an external programming interface also has significant advantages and disadvantages. The first advantage is that access to source code is not required and hence most of the issues associated with accessing source code are removed. The second advantage is sharing data via memory is fast and does not require the utilization of network resources. The third advantage is related to the first, DLLs can be updated without requiring applications to be recompiled or relinked. The fourth advantage of using a DLL is similar to the previous two strategies in that data structure and transport for the most part, is at the discretion of the developer. The first disadvantage of this method is that it involves model modification requiring either the programmer to learn about the model or the modeler to learn about HLA to make necessary function calls. The second disadvantage is DLLs may require significant development effort to provide control over the simulation clock.

The final strategy for interoperability is to couple the engineering application with the RTI via a gateway program. A gateway program acts a data bridge. Unlike a DLL, this is a separate executable. Using this strategy the gateway software can communicate with the RTI per the API specification, but communicates with the engineering application by files, ports, pipes, or networks [4]. Under most circumstances, this is the

least desirable strategy. This strategy puts significant demands on both programmers and modelers. The only obvious advantage of this approach is that it does not require access to source code, external code, or the existence of an API.

The previous paragraphs suggest the implications of the differing programmer's perspective strategies on a model developer. From the user or modelers point of view there is the explicit and implicit approach. In the explicit approach, a library provides users with functions they call from their models. This approach requires a limited amount of knowledge and awareness of HLA by the model developer. In the implicit approach, HLA functionality is hidden from the model developer. This means the functionality must all be handled automatically.

The previously mentioned strategies assume that programmers and modelers belong to different communities, which is not necessarily true. This thesis research does not make that same assumption. Another characterization of approaching integration might be to focus the perspectives on internal versus external model application changes. It is important to consider each one of these strategies when evaluating a simulation package for an approach to interoperability. Also worthy of note is that some circumstances may dictate using a combination of strategies. The advantages and disadvantages associated with the four integration strategies all have three common themes:

1. How is data structured and transferred in and out of the engineering application?
2. How is time managed in the engineering application such that data are delivered and received appropriately?

3. How is the data represented in the FOM such that it is consistent with the conceptual model?

Once the federate designer understands resource constraints and options available for interfacing an application with the RTI, a match must be made between the conceptual model and the constraints imposed by the application.

Data Structure & Data Transfer

How data are structured and transferred in and out of the "engineering" application is critical for interoperability. Often simulation tools permit the import/export (I/O) of data and, as mentioned in Section 3.2, there are a number of methods available to communicate with the RTI. The main methods are: implementation in source code; the use of an external programming interface; or coupling using a gateway program [3] as depicted in Figure 4.

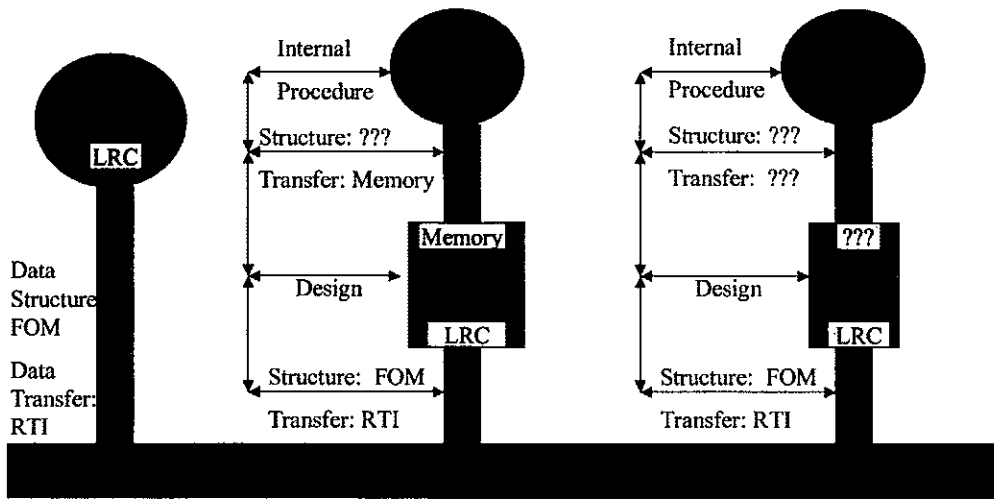

Figure 4. Application Interface Design

Many commercial simulation packages are capable of I/O but often have a specific format and method of opening files for reading and writing data. When directly modifying source code or an external interface, data structure is not particularly

important because a component of the RTI can be instantiated in the application. This instantiated RTI component can communicate directly with the RTI executable (RTIExec) and is commonly referred to as the Local RTI Component (LRC). In instances where creating an external interface or gateway is necessitated or chosen by the federate designer, it is important to evaluate the applications I/O utilities. As a federate designer, there are a number of implementation choices with regard to the FOM, the API design, the applications internal control mechanisms, and data structure and data transfer. The data structure and data transfer out of the engineering application is often the most constraining and limiting factor, which is why evaluation is so important.

From the perspective of the federate designer, the following questions should be considered, and potentially documented in large projects, when evaluating a particular application for integration.

1. What are the possible formats for the inputted and outputted data?
2. Does the data need to be stored, and if so where can this data be stored?
3. Once the data is received, how does it need to be manipulated?
4. How long does it take to input, output, and manipulate the data?
5. Do the data manipulations tax system resources and if so roughly how much?

The answers to these questions will start to define the requirements of the gateway or API. Also noteworthy is that some simulation packages allow C++, visual basic, or macros (OLE) to be imbedded in model code. In some circumstances, these tools are often nearly as powerful as having access to source code.

Time Management

Once an evaluation has been done establishing that data can be exchanged with the legacy model, the next most critical issue is evaluating the necessity and ability to control time within the application. If there is not a requirement to manage time then, the federation designer proceeds to the task of creating an API or Gateway. The API or gateway can then take the output data and communicate to the RTIExec as depicted in Figure 4. An example of a federate that does not require time manipulation is an observer or data collector. Often federations have 3D visualizations such as an observer of battlefield activity, in the NVE. These types of applications may only require update information and have no impact on the simulation outcome or statistical analysis. If a requirement exists to manage time, then there are a number of alternatives available to achieve interoperability. Two techniques for managing time were investigated as part of this research.

External (RTI) Time Management

The first technique is to use the RTI time management services as "intended". Federates can be designated as regulated, constrained, neither or both. Regulating federates regulate the process of time of constrained federates. Basic terminology:

1. Regulating - A regulating federate declares itself to be regulating and is capable of generating time stamp ordered (TSO) events. A regulating federate is capable of controlling the progress in time of federates that are designated as constrained.
2. Lookahead - Regulating federates establish lookahead values, the regulating federates promises that TSO events will occur equal to and no earlier than

"$t_{current} + t_{lookahead}$". This promise ensures federates can process information in "$t_{lookahead}$" without missing external status information.

3. TSO Events – A TSO event is an event with a time stamp. A TSO event must occur at time "$t_{current} + t_{lookahead}$" or greater, but not necessarily in order. For example, a regulating federate could send an event at "$t_{current} + t_{lookahead} +10$" followed by "$t_{current} + t_{lookahead} + 2$". Constrained federates are then responsible for ordering the TSO events.
4. Constrained – A constrained federate declares itself to be constrained and is capable of receiving TSO events. Regulating federates control the progress of time in constrained federates. Also, constrained federates process events in time stamp order.
5. Lower Bound Time Stamp (LBTS) - The LBTS specifies the earliest possible time stamp ordered event the federate can receive. Constrained federates have an associated LBTS [12].

HLA provides the basic tools necessary to deal with time management issues.

Internal Time Mechanisms - The second alternative investigated is establishing a method for controlling progression of the simulation clock internal to the simulation application. This technique makes use of the computer's internal clock to "regulate" model time and relies on a procedure, or federation agreement, to maintain synchronization. The advantage to this method is that it requires a minimal amount of coding and interaction with the RTI. The disadvantage of this method is that it requires simulation operator coordination and is extremely inflexible.

Other Time Management Strategies - There are a number of alternative methods for managing time in a federation. One other option is to use combinations of HLA interactions/services and internal mechanisms. An example of this approach occurred in Fleet Battle Experiment – India (FBE – I). For this experiment, the time management service was removed from the RTI reducing bandwidth and increasing entity count. In this experiment the system clock and interactions were used to synchronize activities. These strategies, although feasible, are complex and, in the case of Fleet Battle Experiment (FBE), modifications were made to the RTI.

The Federation Object Model (FOM) & Interfacing with the RTI

HLA provides numerous methods for disseminating and manipulating FOM information in the federation and, because of this flexibility, it is the last issue addressed prior to creating an interface. Again, there are a number of approaches available. For this research, three alternatives were identified and one was investigated. The three alternatives deal with how and where processes in the engineering application are represented. Objects can be "simulated" in the gateway/interface and then manipulated using HLA's object management and attributes. As a second alternative, objects can be stimulated by the process model and simulated "in" the RTI with declaration management and interactions. The third alternative is to modify the simulation model to represent an object oriented application. This is depicted in Figure 4 by application A. Any of the three alternatives can be used when generating an enterprise model and applicable FOM; however, certain strategies are more efficient, inherently compatible with HLA, and minimize bandwidth.

Using Attributes and Interactions - Attributes and interactions are generated as methods in the LRC and are almost identical procedures. In both cases, an Attribute Handle Value Pair Set (AHVPS) is created and used in the HLA structure to share information. When building an AHVPS, the federate is responsible for any data marshalling (encoding) and the LRC "knows" nothing about data content. This places responsibility on the programmer for writing the methods in the LRC, which is a source of flexibility in managing data (I/O) and means code implementation is similar for both cases [13]. The difference is in how the RTI deals with attributes compared to interactions. Although interactions are constructed in a similar fashion to attribute updates, it must be emphasized that objects and their attributes persist while interactions do not. Each interaction is constructed, sent, and then forgotten. Interaction recipients receive, decode, and apply the interaction. The critical distinction between interactions and attributes is bandwidth and the FOM. Due to the fact that attributes of objects persist, all parameters of an object instance are periodically bundled and transmitted. Interactions, on the other hand, are only sporadically transmitted individually.

Objects "Simulated" in the Gateway

Simulating objects in the gateway presents a number of challenges and potential advantages. This scenario is depicted by application B in Figure 6. The challenge lies in creating a gateway or external interface that accurately exchanges "process information" to represent "object information" (as well as the opposite). The advantage to this approach is that the HLA management services can be used as "designed." To the federation designer, because all the manipulations and translations for application B are

done behind the gateway, applications A and B appear identical. The bandwidth required and the SOMs also are identical.

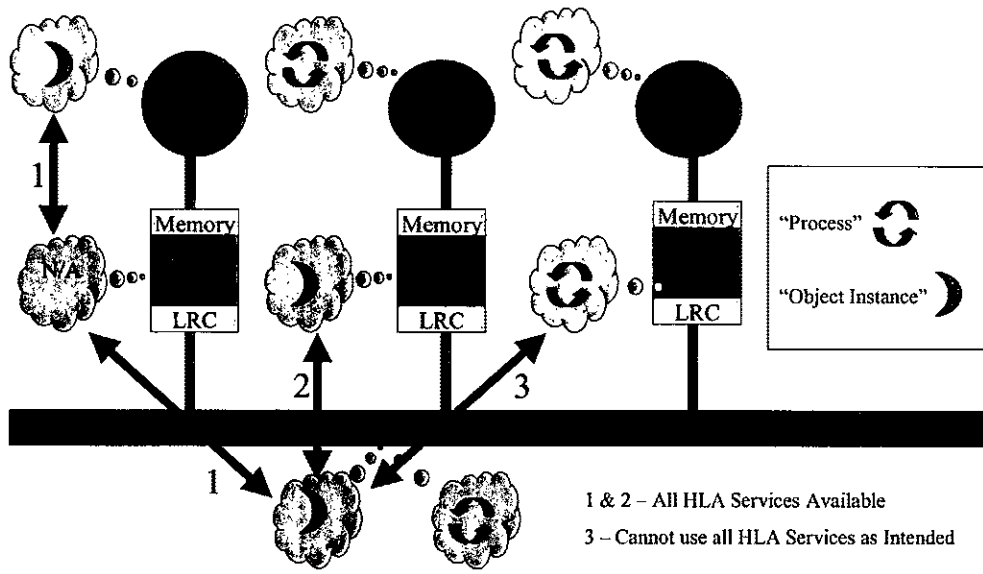

Figure 5. Interfaces Representing Objects and Process

Using this implementation, services like ownership management could be used. Ownership management allows the distribution of attribute responsibility that consists of maintaining updates and deleting instances. Ownership management adds an additional dimension of flexibility because it allows objects to persist after the creating or managing federates cease to exist. This reflects back to the concept of autonomous simulator nodes. If a federate crashes, another federate can assume the responsibility for maintaining the objects. A potential negative aspect is ownership exchange can be quite complex.

Reflected Objects "Simulated" in the RTI - Stimulating reflected objects also offers a number of challenges and advantages. This scenario is depicted by application C in Figure 5. Again, a challenge exists in creating a gateway or external interface that accurately exchanges "process information." However, using this approach, the gateway

is not required to represent "object information." The advantage of this method is process information can be exchanged as interactions. Publishing and subscribing to interactions is more straightforward than registering interests in object classes, which is required when using attributes. The disadvantages to this approach are the diminished capacity of HLA management services and the approach is not transparent to the federation designer. As illustrated in Figure 5, because application C does not possess the same concept of an object as applications A & B and interactions are used rather than attributes, application C will have a differing SOM and bandwidth requirements.

4. EXAMPLE INTEGRATION USING THE FEDEP

This section will step through two sample projects. For ease of reference the two projects are generically referred to as "Project A" and "Project B". The organization of this section is to execute the FEDEP process to accomplish Project A; at the same time, commentary and observations about Project B are added. The reason for this approach is to cover in sufficient detail all of the major steps in the FEDEP process while capturing some of the intricacies exemplified by executing the process as an individual and as a large development team participating in a CAX. As a solo developer in Project A, it is necessary to play the role of many of the stakeholders, such as the customer, chief engineer/federation architect, and developer. Solo-execution greatly simplifies a number of the FEDEP steps by removing the challenges introduced by multiple stakeholders. Commentary on Project B provides insight into execution of a large-scale integration effort as part of a large development team.

Project A, was a proof of concept prototype that was developed to build an enterprise simulation using HLA. The purpose of building the prototype was to explore how various features of the HLA and RTI could be used. Project A, had a single software developer and modeler. The original intention of the project was to link VMASC projects. VMASC had established traffic modeling algorithms and expertise using Arena in execution of the Busch Gardens Traffic Modeling Project. VMASC also, had been significantly involved with HLA research. The legacy Busch Gardens Models/Algorithms and HLA were well matched capabilities for exploring using HLA to integrate simulations for OOTW. Project B, in contrast, was a much larger effort.

Project B, was executed as a model and software developer in a large team preparing for FBE – J.

In order to create the federation, the steps described in the HLA Federation Development and Execution Process (FEDEP), pictured in Figure 6, are employed. A large amount of literature exists in reference to executing the FEDEP; however, very little literature exists that specifically addresses how to create allocated federates. The HLA FEDEP Process Model simply states:

> "In some cases (for non-HLA compliant federates) it may even be necessary to develop an HLA interface for the federate. In this situation, the federate must consider both the resource (e.g., time, cost) constraints of the immediate application as well as longer-term reuse issues in deciding the best overall strategy for completing the federate interface" [17].

Hence, the majority of the interesting and relevant portions of this project occur in the Design Federation Block.

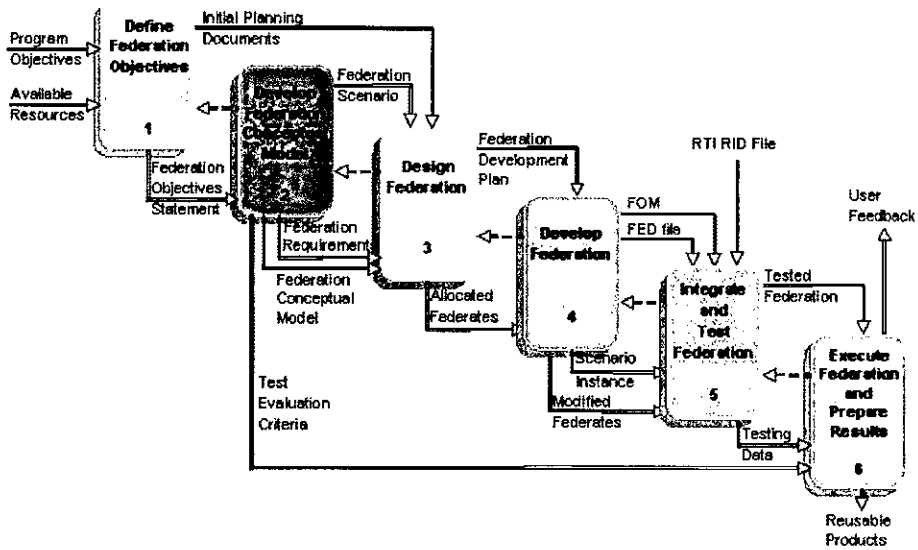

Figure 6. FEDEP

4.1 Define Federation Objectives

The Define Federation Objectives block is the first block in the FEDEP process. This block requires two inputs, the program objectives and the available resources. The outputs of this block are: the initial planning documents and a federation objectives statement. These two items are inputs to the Develop Federation Conceptual Model block. Two activities, identify needs and develop objectives, are the process steps used to transform the program objectives and a notion of the available resources into the initial planning documents and federation objectives statement.

The primary purpose of the Identify Needs activity is to develop a clear understanding of the problem to be addressed by the federation. It includes, at a minimum, a high level description of critical systems of interest, an indication of fidelity and behaviors for simulated entities, key events to be represented in the scenario, and output data requirements. These data are used in conjunction with available resources to create a needs statement. The needs statement identifies the resources necessary to support the federation development, such as funding, personnel, tools, and facilities. In addition, any known constraints, such as due dates and security requirements, also are identified. Once the Needs Statement is produced, it becomes an input to the Develop Objectives activity.

The purpose of the Develop Objectives activity is to transform the Needs Statement into Federation Objectives. The Federation Objectives are the foundation for translating sponsor requirements and expectations to more measurable federation goals. Early assessment of feasibility and risk is also done at this point.

Program Objectives (Input)

This portion of the project was trivial for Project A, but becomes increasingly more difficult as the number of decision makers grows. For Project A, the programs objectives are to: first, create a federation of simulation models; and second, look at the impact of building an arena, the Ted Constant Convocation Center, on traffic flow along Hampton Boulevard.

In general, when a single individual is not playing the role of all the stakeholders, this is a more difficult process. Often in large simulation projects, a team that consists of many players with differing agendas, defines the program objectives. In Project B, the team consists of government representatives, contracts representatives, program managers, peer reviewers, and contractors. In this type of highly complex environment, the NATO COBP (Code of Best Practices) suggests the creation of a Roles Diagram. A Roles Diagram visually represents project participants and their relationships. A well mapped out understanding of the relationships of stakeholders will help define/refine objectives based on the target audience [21].

Available Resources (Input)

Three major types of resources were identified for this thesis research, computer hardware, computer software, and personnel. In project A, as the project manager of a small project, estimating capabilities based on resources was a simple task. In this project, VMASC (Virginia Modeling Analysis and Simulation Center) provided or facilitated acquisition of required hardware and software. One person was accessible, and as such, both the skill set and availability of resources were well known. Subsequently, planning and estimating risk was an insignificant effort. The major

constraint identified was personnel. This constraint dictated the exercise should be achievable with limited C++ programming experience. In larger projects, this becomes much more difficult because there is less understanding of the capabilities, availability and commitment of resources. In this case, it is critical to employ risk mitigation strategies. In larger projects, the physical space available for personnel and equipment also may play a significant role.

Initial Planning Documents (Output)

For Project A, the planning documents were a rough schedule, notes, and word documents. There were a number of additional resources available for program management. In Project A, I was the only available resource so a rough schedule and notes were sufficient.

In Project B, this becomes significantly more complex. Once the resources are evaluated and compared to the program objectives, the initial planning documents and federation objectives statement are generated. For Project B the initial planning documents consist of a study plan, program management plan, a product list, and a work breakdown structure (WBS). In general, the planning documents are a function of the size of the project and culture of the organization. In Project B, the governing documents were the WARCON IDEF/RPG, NATO COBP [21], IEEE 12207 [22], and the DMSO VV&A RPG [23]. There are a number of other process and planning tools available.

Federation Objectives Statement (Output)

The federation objectives statement generally is a formal document because it serves as a contract that is referenced continually during the development process. Development of this document requires close collaboration between the federation

sponsor and development team because, upon completion, it ensures the objectives of the needs statement are implementable and met. The federation objectives statement includes:

- A prioritized list of measurable objectives for the federation;
- A high-level description of key federation characteristics (repeatability, portability, time management approach, etc.);
- A federation development plan showing an approximate schedule and major milestones;
- Estimates of needed equipment, facilities and data;
- Identification of security needs; and
- A configuration management plan.

Project A had a common federation sponsor and development team, and as a result the federation objectives statement directly correlated to the program objectives. In Project B, and larger projects in general, this is the first attempt to bound or scope the problem based on resource constraints.

4.2 Develop the Federation Conceptual Model

This block of the FEDEP requires one input, the Federation Objectives Statement, which was defined earlier. The outputs of this block are a federation scenario, the federation requirements, and the federation conceptual model. In the portion of the process called develop conceptual model the federation objectives are evaluated to generate scenarios, federation requirements, the conceptual model, and the test evaluation criteria. Executing the develop scenario, perform conceptual analysis, and develop federation requirements activities generates the outputs for this process block.

The major activities that occur in the develop scenario process are to conduct a search for existing scenarios, possibly in the DMSO scenario database, and attempt to match them to the federation objectives and the conceptual model. If acceptable scenarios do not exist, new ones must be generated.

The second activity performed in the develop federation conceptual model block is the conceptual analysis. The conceptual analysis activity is the process where the federation development team produces a conceptual representation of the problem space identified in the federation objectives statement.

The Develop Federation Requirements activity is accomplished concurrently with the conceptual analysis. This activity is where the federation requirements are generated. The requirements should be directly testable and provide implementation level guidance necessary to design and develop the federation. This activity also should explicitly determine the level of fidelity, so fidelity requirements can be considered when selecting participating federates.

Federation Scenario (Output)

The federation scenario is an initial description of the NVE and often broadly indicates the types and numbers of major entities to be represented in the federation. The scenario also provides a functional description of the capabilities, behaviors, and relationships between major entities over time and is a specification of relevant operational conditions. Scenarios can be presented as graphical illustrations, event-trace diagrams, text descriptions, and a number of other formats. Scenarios create a link to reality, explanation or insight into assumptions, and input into the current state of state variables in the simulation.

In Project A, the federation objectives statement specified evaluating the impact of the Ted Constant Convocation Center on traffic flow. As a result of the objectives statement, the initial scenario was to evaluate vehicle congestion between two intersections on a Thursday evening, with clear weather, after an Old Dominion Lady Monarchs basketball game. This initial scenario was chosen because it was considered a realistic test. The graphical representation of the scenario is illustrated in Figure 7. The development of the initial scenario had an expected outcome, the Federation Scenario, as well as an unexpected one.

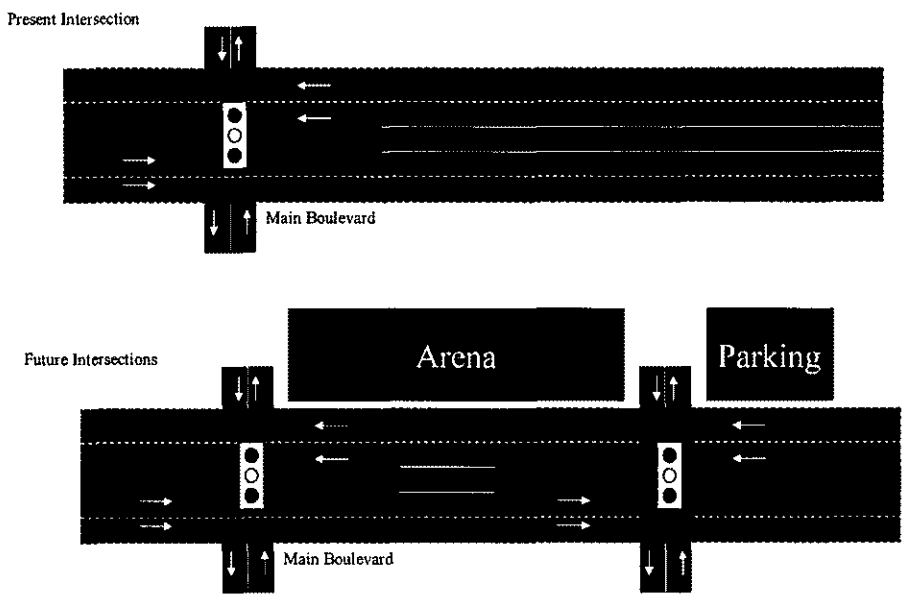

Figure 7. Federation Scenario

Developing the scenario unexpectedly identified data collection requirements. Creating the scenario made it apparent that to do analysis, Virginia Department of Transportation (VDOT) data or observation of current traffic conditions and vehicle quantities corresponding to event attendance would be required. For this thesis, rough estimates were used.

Federation Conceptual Model (Output)

The conceptual model emphasizes objects and their relationships with each other [24]. A number of strategies and tools are available for building conceptual models [25]. For Project A, the conceptual model, pictured in Figure 8, was created in UML (Universal Modeling Language) using Visio® Professional 5.0b. UML was selected because it is OO and has wide acceptance in the software development community. Visio ® was selected due to its suitability and availability. An interesting correlation was made while creating the conceptual model. In BOS, simulation doctrine and tactics have a significant impact on the conceptual analysis and subsequently the conceptual model. In transportation simulations, traffic regulations can replace military doctrine.

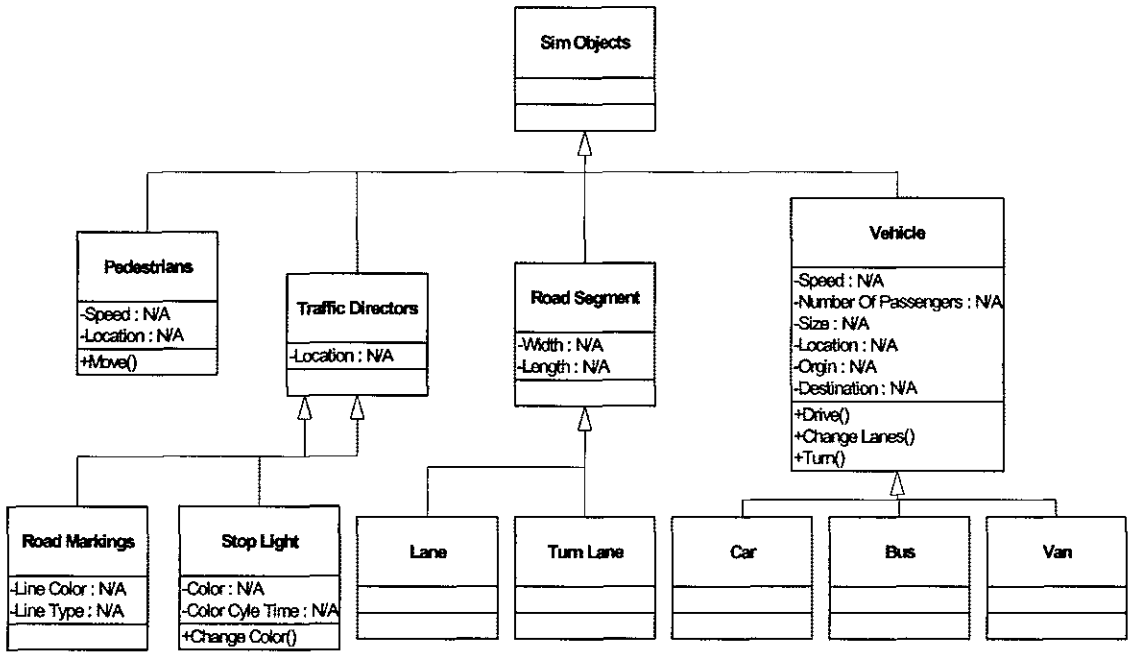

Figure 8. Conceptual Model

Federation Requirements (Output)

The federation requirements are based on the federation objectives and provide implementation level guidance. These requirements should be testable and the fidelity of federation participants should be addressed explicitly. The requirements also should address programmatic or technical constraints on the federation implementation.

In Project A, a formal federation document was not deemed necessary because the scenario, federation objectives, and conceptual model provided sufficient guidance. The level of fidelity, conversely, was specifically addressed. The scenario and conceptual model identified the potential to differentiate between trucks, buses, cars, vans, motorcycles and pedestrians. The level of fidelity selected for implementation, however was that of a generic vehicle. This decision interestingly demonstrated the value of the FEDEP. As discussed in Section 4.2, the scenario exposed potential data collection requirements in regards to traffic patterns and flow rates. The original notion was that data would originate from three sources, ODU, VDOT, and observation. Logistical traffic information would be acquired from VDOT and observation, while Convocation Center blueprints and basketball game attendance figures would be provided by ODU. Upon further investigation, it became apparent that acquiring logistical information from VDOT would interfere with the project schedule. The lack of relevant input data regarding vehicle type made it apparent that division based on type would have no statistical relevance while significantly increasing workload. Execution of the FEDEP generated a schedule, planning documents, and an objectives statement, which made it readily apparent that modeling a generic vehicle was sufficient.

4.3 Design Federation

The purpose of this step is to identify, evaluate, and select federates. After federates are selected, functionality is allocated and a detailed plan for development and implementation is produced. Three activities are specified by the FEDEP, select federates, allocate functionality, and prepare the plan.

The purpose of the select federates activity is to determine the suitability of federates to be members of the federation. In some instances the federation sponsor may predetermine federation membership. Identifying federates may consist of reviewing existing SOMs in the HLA object model library (OML), exploring other sources for models, or making a determination to build new models.

Once all federates are identified, the responsibility of representing the entities and actions in the conceptual model are allocated to the federates. This list of allocated federates then is used as an input in the prepare plan activity.

The final activity in the Federation Design is to develop a coordinated plan for development test and execution of the federation. This plan identifies the software tools that will be used to support the federation life cycle. Examples are the RTI Version, federation runtime tools, CASE tools, configuration management strategy, VV&A, and testing. If the federation contains stochastic variability, the plan also should include an experimental design, variance reduction techniques to be applied, and a determination of the number of replications to achieve a desired confidence interval. These agreements should be included in a detailed work plan and documented for later reference and possible reuse in other federations.

Allocated Federates (Output)

There are a number of methods for distributing the responsibility for entities and actions among the participating federates. Two possibilities are entities can be distributed to federates based on type, or entities can be distributed one per federate for man in the loop exercises. In Project A, entities and actions were distributed based on geography. Figure 9 illustrates a natural geographic separation point, as the two intersections are almost identical. Reviewing existing SOMs in the HLA OML was determined to be of little value. VMASC, however, had traffic algorithms that were generated in conjunction with the Busch Gardens Traffic Simulation project that exhibited strong potential for reuse.

Federation Development Plan (Output)

For Project A, the federation development plan was informal. RTI Version 1.3 NG was selected based on availability. Although not specified, Microsoft® Word 2000 (Version 9.0.3821 SR-1) was selected as the tool for documents, Visio® Professional 5.0b was selected for UML, and Microsoft ® PowerPoint® 2000 9.0.2716 was selected for diagrams. No other tools were specified or identified during this portion of the process. Configuration management consisted of an *ad hoc* file saving and naming convention. VV&A and testing were to be done through observation as required. Statistical rigor was considered unnecessary for proof of concept.

4.4 Develop Federation

The purpose of this step is to develop the FOM, modify federates as necessary, and prepare the federation for integration and testing. The output of this block is the FOM, FED file, a scenario instance, and the modified federates. The Develop Federation

step consists of three activities to produce products. Develop FOM, Establish Federation Agreements and Implement Federate Modifications.

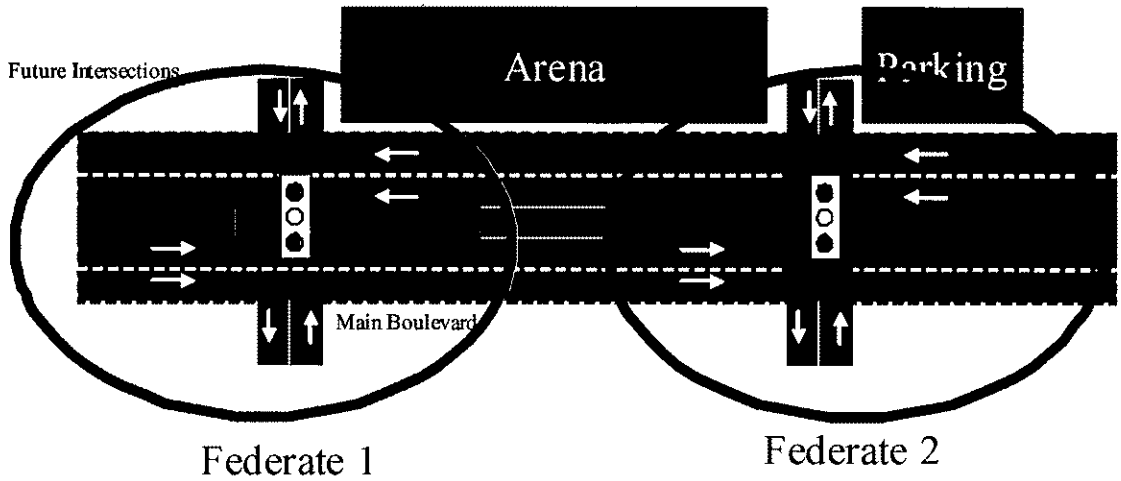

Figure 9. Allocated Federates

The FOM is developed to support data exchange among federates. There are four major approaches to creating the FOM: construct the FOM from the bottom up using the OMDD (Object Model Data Dictionary); merge the SOM's of all the participating federates and remove aspects that are not applicable to the domain of interest; begin with the SOM that most closely represents the desired FOM and merge the aspects of the other SOMS to fully represent the domain, or begin with an existing FOM that is similar and modify and augment as necessary to represent the desired domain. Regardless of the approach chosen, use of automation tools are strongly encouraged. One such tool is the OMDT. The OMDT is available free of charge from DMSO but has received a number of poor reviews from the user community.

The second major activity of this process block is to Establish Federation Agreements. The FOM defines and documents the set of data that is exchanged among federates; however; it is not sufficient. Other federation agreements are made to ensure

consistent interaction of federation objects with data. The FOM, for example, technically defines the communications necessary to execute a federation synchronization but the federation agreement would specify when and how it is appropriate. A synchronization point, commonly referred to as a sync point, is a HLA provided method that queries a targeted set of federates and is sometimes used to prevent the advancement of simulation time until all targeted federates are synchronized. Federation agreements are also used to ensure "fair fights" by establishing a consistent federation-wide view of environmental features. An example is depicted in Figure 10. The federation agreement ensures that both aircrafts pilots not only receive information about the cloud, but that the cloud also is perceived consistently; that is, aircraft A can shoot aircraft B but B can not shoot A because the federates respond to identical data in a differing manner.

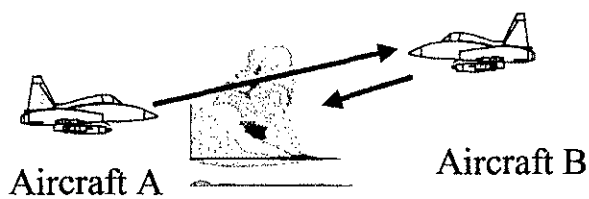

Figure 10. Fair Fight Federation Agreement

The Implement Federate Modifications activity is the third and final activity of the Develop Federation process block. This activity is one of the primary focuses for this thesis. The purpose of this activity is to implement all the modifications necessary to ensure the federates can represent the objects and behaviors allocated from the conceptual model.

The Federation Object Model (Output)

Before the federation execution can exist, it needs to be defined to the RTI by being associated with a FOM. The FOM is created using the standard Object Model template (OMT) that prescribes the allowed structure of every FOM under the HLA rules.

Determining a FOM for Project A, as depicted in Figure 11, was a relatively simple task. The approach taken was to modify an existing sample FOM available in the RTI sample code. In Project B, as part of a large exercise determining the FOM was a significantly more complex task. Project B, utilized a modified version of the Real-time Platform Reference (RPR) FOM. The RPR FOM was used because DIS simulations composed a large number of the participants. The RPR FOM contains a representation of many of the DIS PDU's allowing DIS simulations to participate in HLA federations.

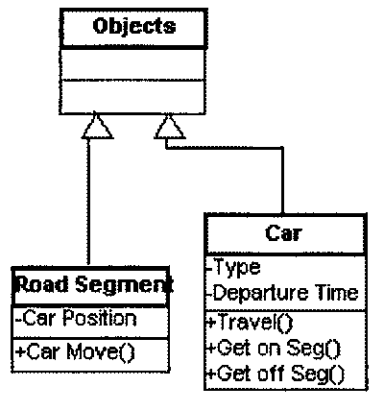

Figure 11. FOM Class Diagram

In general, larger exercises result in larger FOMs simply due to the larger quantity of SOMs involved.

Federation Execution Data (Output)

The Federation Execution Data (FED) file supplies the RTI with federation execution details during new federation creation. The FED file defines routing spaces, objects and interactions [29]. The federation execution data was not modified for Project A.

4.5 Integrate & Test Federation

The purpose of this step is to plan the federation execution, establish interconnectivity between federates, and then test the federation. This step consists of three activities: Plan Execution, Integrate Federation, and Test Federation. The output of this step is the testing data and the tested federation.

The purpose of the Plan Execution activity is to define and develop the full set of information required for federation execution. This includes refining test and VV&A plans. The primary task for this activity, however, is to document information in the FEPW (Federation Execution Planners Workbook). The FEPW contains host and network requirements. The completed workbook defines all the necessary information for a federation developer to test and operate a federation. If the federation is to be secure, this is where the security test and evaluation plan is developed [30]. Lastly, the RID (RTI Initialization Data) file may be modified in this activity for the sake of performance enhancements.

The Integrate Federation activity is where all the federation elements are brought together in a unifying operating environment. This requires all hardware and software is properly installed and interconnected in a configuration that supports the FOM interchange requirements and federation agreements.

The last activity performed in the Integrate and Test Federation step is the actual testing of the federation. The purpose of the Test Federation activity is to test that all the federation participants can interoperate. Three levels of testing are defined for HLA applications.

1. Federate Testing: In this activity, each federate is tested to ensure the federate software is correctly implemented as documented in the FOM, FEPW and any other federation operating agreements.
2. Integration Testing: The federation is tested as an integrated whole to verify basic interoperability. This test is primarily concerned with the ability of federates to interact with the RTI and exchange the data specified in the FOM.
3. Federation Testing: The ability of the federation to interoperate to the degree necessary to achieve the federation objectives is tested.

Tested Federation (Output)

In Project A, the federation was tested in conjunction with the modification of the federates. In Project B, the federation was tested in a series of integration events and spirals.

Testing Data (Output)

The testing data are criteria established to indicate when federation execution can commence. DMSO offers a HLA compliance checklist to facilitate this task [29]. In Project A, the federation was tested in conjunction with the modification of the federates. In Project B, a formal federation test document was used to verify all participating federates met the acceptability criteria. For example, experiments were run on federates

to verify they could join and resign from the federation with out adversely affecting other federates.

4.6 Execute Federation & Prepare Results

The purpose of this step of the FEDEP is to execute the federation, process the output, and prepare the results. This step consists of three activities: Execute Federation, Process Output, and Prepare Results. The outputs from this step are a corrective actions report, user feedback, and reusable products.

The purpose of this Execute Federation activity is to execute the simulation exercise with all the participants, generate the simulation data, and achieve the federation objectives. An additional purpose of this activity is to capture federation playbacks and evaluate federation performance.

The second activity of the Execute Federation and Prepare results step is the process output activity. The purpose of the Process Output activity is to process the data collected during execution. This process often requires statistical analysis tools to transform raw data into derived results.

The Prepare Results activity consists of two tasks. Task one involves evaluating the raw and processed output data. Once the evaluation is complete a determination of whether changes and re-execution is required. Task two, is to store any reusable products. Products can be stored in the DMSO Modeling and Simulation Resource Repository (MSRR). Examples of reusable products are the FOM, OMDD entries, federation scenarios, and conceptual models.

Reusable Products (Output)

Project A was a proof of concept exercise and, as a result, none of products were considered reusable. More commonly, as with Project B, the FOM, FED, RID and a number of other reusable products are stored in the MSRR [32].

User Feedback (Output)

Project A was a proof of concept exercise user feedback was not required. In Project B, and larger projects in general, user feedback may take the form of an after action review (AAR). The AAR is a common method for collecting lessons learned and user feedback.

5. PROCESS EXECUTION

This section continues to step through Project A executing the Modify Federate Process depicted in Figure 3. The Modify Federate Process links the FEDEPs Develop Federation, Step 4, to the Integrate and Test Federation, Step 5 depicted in Figure 6. The organization of this chapter is to describe setup requirements followed by a detailed description of executing the Modify Federate Process.

5.1 Setup Information

Available resources are one of the inputs to the Define Federation Objectives step of the FEDEP. For this particular project, a significant amount of effort was invested increasing the skill set of the programming/modeling resource by researching and learning the basic skills necessary to create a federation. The basic skill set upon onset of the project were less than a year programming with C++, a strong background in computers, one year experience with Modeling & Simulation, less than a year experience with Arena, and a brief two week exposure to HLA in a graduate M&S course. A number of skills for this project were learned while executing the process. The skills that are thought to be needed only on a case-by-case basis will be addressed while describing the execution of the Modify Federate Process. The following reference materials are recommended prerequisites:

1. The DMSO and Simulation Interoperability and Standards Organization (SISO) web sites [26].
2. The FEDEP [17].
3. The HLA Run Time Infrastructure Programming Guide [13].

4. A Network Administration guide. For Project A, Network+ by CompTIA was used [27]. Recommended topics of interest are configuring DHCP, DNS, configuring IP Addresses, subnet masks, pinging, and a general description of network components.
5. RTI-NG 1.3v3.2 Installation Guide [28].
6. A programming language reference. For Project A, C++ Primer Plus was used as a reference for C++ [33]. Visual C++5 was identified as a reference for the Visual Studio 6.0 IDE [34].
7. Hands-On Practicum Programming with the Run Time Infrastructure [35].

The following hardware, software, and network environment were established as a learning, development, and execution environment. When executing the FEDEP as part of a large federation, these quite possibly could be separate environments that are established as needed. In Project B, federation teams often had small development environments, medium sized testing environments, and finally the full environment for execution.

Hardware

A high-level hardware list is generated as part of Activity 1.1, Identify Needs (of the FEDEP) and then the actual hardware is acquired and configured as part of Activity 5.2, Integrate Federation. In Project A, the hardware list was specified by availability. The hardware available for development was also the hardware available for execution, thus greatly simplifying Activities 1.1 and 5.2 of the FEDEP. The computers used in Project A were:

- Dell® computer with a 400 MHz processor, 64 MB RAM, 3.5" 1.44MB Floppy Dive, 10/100 Mbps Ethernet NIC (Network Interface Card), 4X CD-ROM Drive.
- Gateway® Computer with a 800 MHz processor, 128 MB RAM, 3.5" 1.44MB Floppy Dive, 10/100 Mbps Ethernet NIC, 24X CD-ROM Drive.

Software

A high level notion of software required to support federation execution is generated as part of Activity 1.1, Identify Needs (of the FEDEP). This list becomes refined in Activity 3.3, Prepare Plan, which outputs the Federation Development Plan. The software for execution is acquired and configured as part of Activity 5.2, Integrate Federation. In Project A, the software list again was partially specified by availability. The software list is below is distributed between the two CPUs used. For this project the Dell CPU was used as the development computer and the Gateway was used primarily for federate execution. This distinction is noteworthy because, often for federation executions, only the hardware and software necessary to run the federate is theoretically required. The relevant software is as follows:

- Dell (Development & Execution Computer) – Windows NT 4.0 with Service Pack 6 as the operating system (OS), RTI Version 1.3 NG for Windows NT (for this exercise the standard RTI 1.3NG was used. This software is freely available for download from the DMSO website and required no modification for use in this federation), Microsoft® Word 2000 Version 9.0.3821 SR-1, Visio® Professional 5.0b, Microsoft ® PowerPoint® 2000 9.0.2716, Microsoft Visual Studio 6.0 for running, decoding and debugging, Microsoft Excel, VNC (Virtual Network

Computing) to view a remote computers desktop via the local PC, and Rockwell ® Arena 5.1.

- Gateway (Execution Computer) – Windows 2000 Professional (Operating System), RTI Version 1.3 NG for Windows 2000 Professional, Microsoft Excel, VNC (Virtual Network Computing) and Rockwell ®Arena 5.1.

The RTI was installed in accordance with the RTI-NG 1.3v3.2 Installation Guide [28]. Careful attention is required when installing the RTI as there are a number of different versions. The RTI has been turned over to the commercial sector from DMSO and as result it is expected that installation will become significantly more user friendly. When installing the RTI for those who are not familiar with environment variables locating a reference is highly recommended. A search in Windows Help may provide sufficient information.

Networking

A high level notion of the network required to support federation execution is generated as part of Activity 1.1, Identify Needs (of the FEDEP). This list becomes refined in Activity 3.3, Prepare Plan, which outputs the Federation Development Plan. The Network for execution is acquired and configured as part of Activity 5.2, Integrate Federation.

In Project A, the VMASC LAN was originally used for execution. Due to the amount of network traffic generated during experimentation the VMASC LAN was replaced by a Linksys Etherfast Workgroup (EZXS55W) 100Mbps, 5 port HUB.

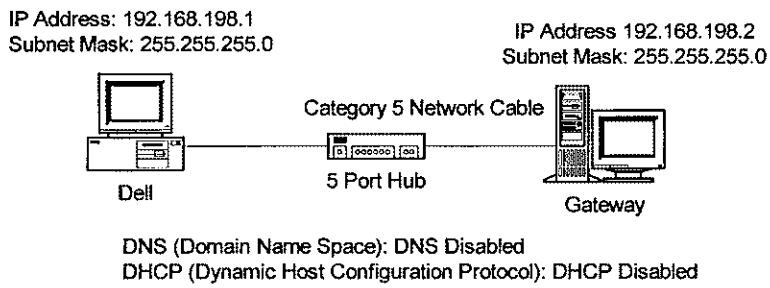

Figure 12. Network Diagram.

Project B, as previously mentioned was executed on a number of LANs of differing configurations. Project B, was executed on the Virginia Advanced Shipbuilding Carrier Integration Center (VASCIC) Simulation Network for development, Joint Forces Command (JFCOM) network for the interoperability tests commonly referred to as Spirals, and via the Secret Defense Research Engineering Network (SDREN) for the experiment.

5.2 Executing the Modify Federate Process

The Modify Federate Process depicted in Figure 3 can be separated, as discussed in Chapter 3, into five major processes assuming the federate is not already HLA compliant. First, an evaluation of the applications accessibility is performed and a strategy is selected based on those findings. Second, if required, an evaluation is performed of the applications I/O utilities and a preliminary determination on the structure and format of the data. Third, an evaluation of simulation time is performed in conjunction with a decision on how to handle simulation time. Fourth, an evaluation of interfacing with the RTI, FOM and the data structure selected is performed. Fifth, the interface is created.

Evaluation and Strategy

Arena is not natively HLA compliant necessitating the execution of the remaining Modify Federate Process. Evaluation of the Arena simulation software identified two of the four strategies mentioned in Section 3.2 as feasible solutions. The Arena source code was not readily available, eliminating the option of modifying the source code. The stipulation of "not readily available" is made because Rockwell Software was not approached with a request for access to the source code. Occasionally companies will negotiate agreements permitting access. Arena does not have an open extensible architecture eliminating the third strategy of creating an external interface. Arena, however, does have an alterable external interface, making this option feasible. Almost every application has a method for data I/O making a gateway nearly always technically feasible although maybe not practical. The primary strategy selected for Project A was to alter the external interface. Creation of a gateway was dismissed temporarily but still feasible as a back up plan.

Familiarity with Arena made this evaluation process and choice of strategy relatively simple. Research of the provided documentation and help files provided the requisite material and insight [36]. Other points of reference are online help, or technical support via phone or the Internet [37]. For Project A, Arena's help file proved to be the most beneficial. The help file indicated that external interfaces in the form of Dynamic Link Libraries (DLL) exist for both FORTRAN and C++. This discovery necessitated further investigation of DLLs. DLLs for beginners was downloaded from the Microsoft website [20].

DLL are files separate from executable applications as pictured in Figure 13. DLLs are called by other programs to perform computations or functions. Dynamic linking, links applications to data libraries at run time. When an application uses a DLL, the operating system loads the DLL into memory, resolves references to functions in the DLL, so that they can be called by the application, and unloads the DLL when it is no longer required [20].

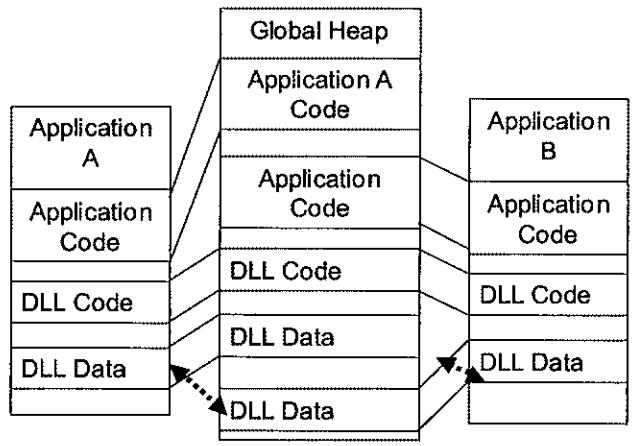

Figure 13. Dynamic Link Library Using Shared Memory

The C++ external interface or Application Program Interface (API) was a natural choice because the LRC and libRTI are written in C++ and this was the programming language with which the resources were most familiar [38]. In larger projects where the programming group and modeling group are separate, close interaction between the two is required to select an alternative.

Evaluation of I/O Utilities & Determination of Data Format

The purpose of this process is to determine a method for communicating between the application and the RTI. In Arena's external interface, there are a number of methods available for data transfer such as shared memory, files and ports [20], but there is a specific format and method of opening files for reading and writing data. This fact is not

intuitive or specifically stated in Arena's user's guide or advanced user's guide, but a significant amount of information was available in the help file.

For this project, four methods were identified and two actually were tested. The four methods identified are data transfer via shared memory using the DLL, reading and writing to files, use of a port, and a hybrid combination. Using shared memory via the DLL was determined to be outside the capabilities of the resources available.

Researching reading and writing to files exposed a number of options and issues. The first concept investigated was to use Arena's internal file manipulation. Arena natively allows modelers to use READ and WRITE. These commands allow the modeler to, input or output, integer or real number values, between Arena and Microsoft Excel. There are a number of problems when trying to use these functions in any type of real time or multiple of real time environment. The read and write functions, open a file read or write to the file and then close the file. The first issue is, opening writing or reading and closing a file is it is very time consuming. This affects two things: First, is the execution of the simulation. The simulation time will not advance until the data transaction is complete slowing down execution. Second, is the significance of time data for analysis. The frequency that Arena can poll the Excel spreadsheet may limit the granularity of time data. The premise is that, in time synchronized systems, if state update information is received every ten minutes, the shortest time increment that has any statistical or analytical significance must be ten minutes. The period of each polling cycle must be longer than the longest read or write process, and subsequently might have an effect on the data collection requirements. The third issue is the format the data is received. The concept investigated for using the read and write commands was to

establish Message Queues in Excel. Each Arena simulation would have a Read Queue and a Write Queue. The simulation would then poll its read queue at some regular interval to read in information. The simulations would write to the Write Queue as necessary and the federate interface would poll the Write Queue. After the Federate interface polled the Write Queue it would then manipulate the data to communicate via a LRC as depicted in Figure 14. The last issue associated with using files is due to the limited set of commands available in Arena. Manipulation of the data in the read and write queues is extremely difficult. For this experiment information was strictly passed through the Federation Interface. This meant the quantity and instant data is placed and removed from the queues is a function of events in the simulation and the polling cycle.

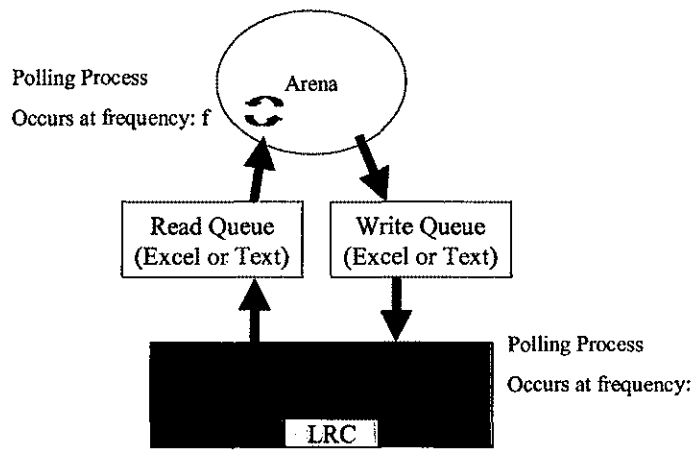

Figure 14. Excel Gateway

Arena's read and write functions are intended to read and write a pre-specified number of characters one line at a time. Parsing an arbitrary number of characters or determining a consistent manner to read, write and erase, although feasible is a significant challenge. This particular approach is probably best avoided if at all possible. If however, this is the only approach available, it is highly recommended that a standardized type, frequency

and quantity of data is passed through the files. Also, manipulation and queuing of the data, if at all possible, should be handled in the federation interface.

The second method investigated and chosen for this project was to utilize ports. A port is essentially an electrical connection through which software can communicate. Ports can be difficult and time consuming to program and as a result a pre-written program was located to manage communications via a port. This is called an inter-process communication (IPC) program. For this particular experiment simple server, userc.c, and simple client, simplec.c, were used as depicted in Figure 15. IPC's are generally used at a minimum as a pair. This software is free to download and is part of the Microsoft source code samples. The IPC program was used to communicate from Arena's API, RTLL.DLL into which the server code was copied, with the Federation Interface into which the client code was copied.

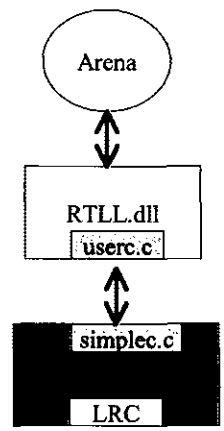

Figure 15. Port Gateway

Arenas external interface can both input and output numerical American Standard Code for Information Interchange (ASCII) characters so a simple data structure was chosen; the first character was the interaction or car type. Cars where separated into three

color coded categories blue, from the convocation center red, normal traffic, and brown, from ODU. The vehicle types were enumerated as 1, 2, and 3 respectively. The next four characters represented the time in minutes as depicted in Figure 16. An assumption was made that the simulation would execute for a day based on the scenario hence the range, of 0000 – 1440 minutes. This also facilitates the data structure by maintaining the time stamp a consistent number of characters

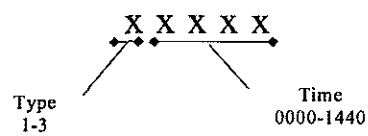

Figure 16. Data Structure

Time Management

The third major process is the evaluation of the necessity of controlling time and if required selecting a strategy. There are two main strategies and a number of variations of these strategies for managing time. The first alternative is to establish a procedural method for managing time, potentially specified in a federation agreement. The essence of this method is that simulations are synchronized manually. An example of this approach is manned simulators. If all the federates agree to advance in real time, or some multiple of real time, an internal control mechanism can be utilized. All that is required is a common understanding of wall clock time. Arena for example has a function CLOCK. The simulation can be stopped and advanced based on the computer clock using if statements. The second approach is to use an external method in conjunction with an internal mechanism to advance time. In this particular instance, the external method is time management. Worthy of note is that for simulations that are only loosely

coupled by time HLAs declaration management service could be utilized to create a time advance interaction that was only utilized occasionally as needed for synchronization. This is part of what was done, in Project B, for FBE to eliminate the need for the time management service.

Time management work was not in the scope of the original proof of concept. Each federation had an internal clock and the clocks were set to run at similar speeds. Additionally, a sync point was used to properly align federate and federation starts points. This method uses the wall clock to synchronize federates without the added message traffic and DLL modification.

RTI Interface

The fifth and final major process is to review the products of the previous steps make a determination of the FOM, time management scheme and data to create the federation interface.

For Project A the 'Foodfight' RTI tutorial exercise was used as a starting point to create a Federation Runtime Interface [35]. The stub contains the basic elements required for the federation as well as useful tutorial information to assist in setting up the interface. The remaining challenge in this process is to integrate simplec.c and create the desired interactions.

Interface Creation & Modified Application

The execution of the FEDEP and the Modify Federate Process resulted in the federation and interface design depicted in Figure 17.

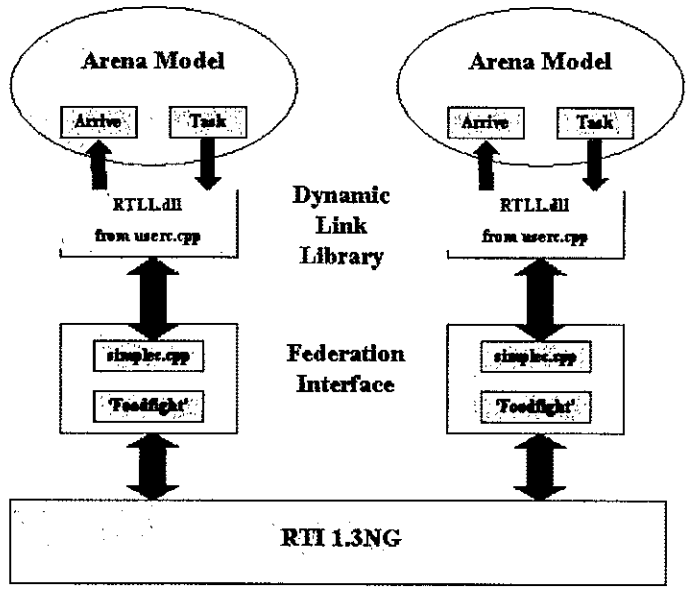

Figure 17. Federation & Interface Design

Figure 17 represents the federation and interface design. Figure 18 below, depicts how Arena, Arena's external interface, the communication software, LRC and RTI interact to create and join a federation execution. The Federation Management Life Cycle is the procedure required to create or join a federation per the Programmers Guide. For Project A this process is mapped to components defined for interoperability. The triggering event in Arena to join the federation, is the initialization of the simulation. An RTLL.dll call is made with enumerated data (1, 5, 7). The data is transferred from RTLL.dll via the socket connection, user.c and simplec.c, to the federation interface. The federation interface resolves the enumerated data into the format agreed upon in the FOM and communicates with the RTI via the LRC.

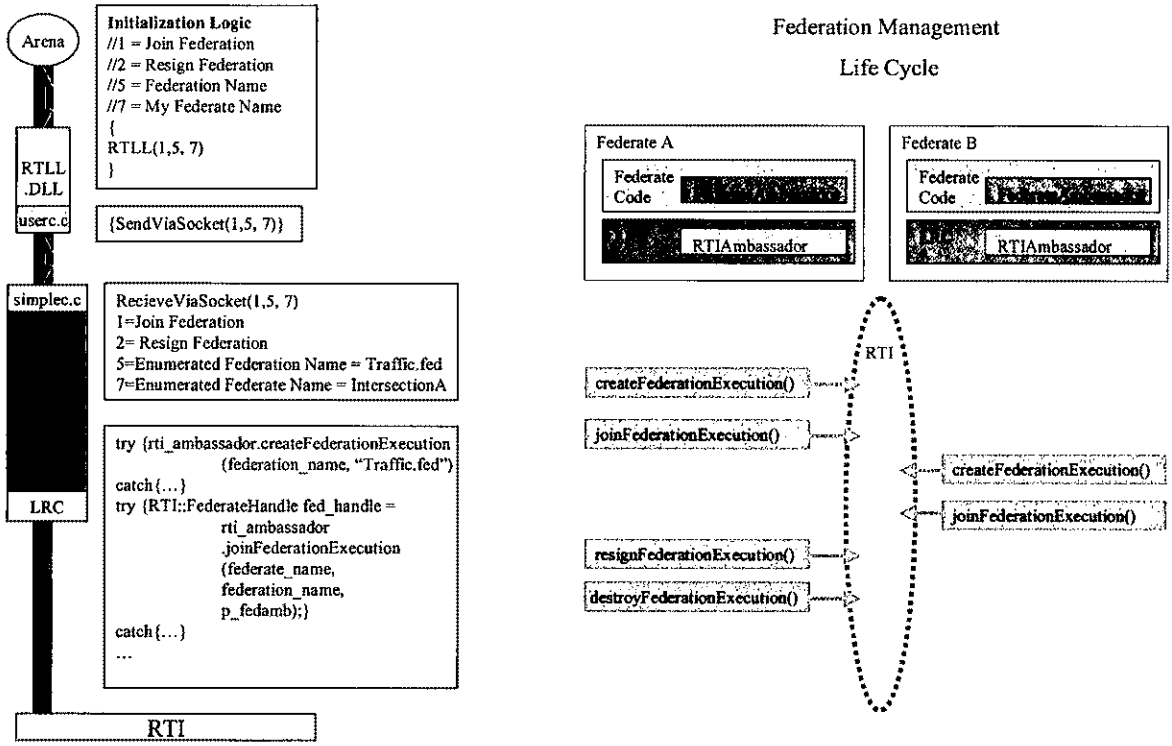

Figure 18. Federation Execution Design and JoinFederationExecution

6. Conclusion

6.1 Important Findings

The aim of this research was to demonstrate that HLA could be employed with common commercial modeling and simulation packages by developing and evaluating a methodology for distribution via HLA. This methodology dictates a need for the ability to distribute simulations that have limited C++, VBA, or various other programming languages accessibility. At this stage a basic methodology and procedure has been specified to evaluate the feasibility of distributing a commercial simulation package. Secondly, the key elements that are necessary to distribute a given commercial package have been identified, and lastly, a proof of concept was accomplished to validate the prescribed methodology. In this procedure a particular emphasis was put on the integration procedure and methodology, the uses of object classes and interactions for enterprise models in the federation object model, the advantages and disadvantages associated with integrating process oriented and distributed interactive models, and critical issues associated with time management.

The organization of this thesis and the content within characterizes the lessons learned. The First Chapter provides perspective into the current modeling and simulation environment and historical context. The Second Chapter establishes a basic knowledge about M&S, enterprise simulations and HLA. The Third Chapter introduces the FOM, time management, data structures and some strategies for harmonizing the complex interrelationship between them. In the fourth chapter the FEDEP process is executed with descriptions and examples of the deliverables. The Fifth Chapter then

explores, at an implementation level, some of the requirements necessary to build an HLA compliant federate/federation.

An enterprise simulation is a simulation which is constructed with a top-down view of a business enterprise and which is intended to serve as a decision support tool for decision makers. This thesis project demonstrated how NSA enablers like HLA can extend the utilization and functionality of enterprise simulations by integrating "best of breed" COTS tools, applications, and simulations. This type of endeavor exposes a number of challenges. Creating enterprise simulations using HLA requires expertise in hardware, software and network engineering. HLA imposes a number of challenges because of time management, object management, and data exchange; however, these challenges are surmountable with sufficient reference material and a process for execution.

6.2 Future Work

This thesis established a basic procedure and methodology for integrating OOTW simulations using HLA. This procedure is just a beginning. Future efforts can build upon this work by:

1. Creating additional enterprise simulation models to expand and test the established process.
2. Creating applications with advanced time management schemes.
3. Constructing a federation of multiple mixed federates.

REFERENCES

[1] Kuakoranta, Timo. "DIS and SIMNET", *Special Course on Networked Virtual Environments Spring 2002,* Spring 2002. Online document at: http://staff.cs.utu.fi/kurssit/spring_2002/.

[2] Mielke, Roland R. "Applications for Enterprise Simulation." *Proceedings of the 1999 Winter Simulation Conference*, Orlando, FL. December 1999.

[3] Fishwick, Paul A. "Simulation Model Design and Execution: Building Digital Worlds." Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[4] Schulze, T., Straburger, S., and Klien, U. "Migration of HLA into the Civil Domains: Solutions and Prototypes for Transportation Applications." *Simulation*, Vol. 73, No. 5, pp 296-303, Nov. 1999.

[5] Erwin, Sandra I., "Video Games Gaining Clout As Military Training Tools." *National Defense*, Nov. 2000. Online document at: http://www.nationaldefensemagazine.org/article.cfm?Id=352.

[6] Stumm, Robert E., and Keener, Rosemarie V. "WARCON Supporting Acquisition Investment Decisions." *Interservice/Industry Training, Simulation and Education Conference Proceedings*, Orlando, FL. May 2001.

[7] Trainer, Joshua P. "Engineering Collaboration on the Desktop A Proof of Concept Prototype." *Proceedings of The Spring 2001 Simulation Interoperability Workshop*, Orlando, FL. March 2001.

[8] Seidel, David W., Testani, Sandra, and Wagner, Ed. "CIMBLE Distributed Team Training via HLA." *Simulation,* Vol. 73, No. 5 pp 304-310, Nov. 1999.

[9] Petty, Mikel D., Windyga, Piotr S. "A High Level Architecture based Medical Simulation System." *Simulation,* Vol. 73, No. 5 pp 281-287, Nov. 1999.

[10] IEEE, "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules." IEEE Std. No. 1516-2000, 2000.

[11] Ziegler, Bernard P., Hall, Steve B., and Sarjoughian, Hessam S. "Exploiting DEVS To Promote Interoperability and Reuse in Lockheed's Corporate Environment." *Simulation,* Vol. 73, No. 5 pp 288-295, Nov. 1999.

[12] IEEE, "IEEE Standard for Distributed Interactive Simulation – Application Protocols IEEE Std. 1278.1-1995." IEEE Std. No. 1278.1-1995, 1995.

[13] Defense Modeling and Simulation Office (DMSO). "High Level Architecture Run-Time Infrastructure RTI 1.3 Next Generation Programmer's Guide, Version 3.2." Sep. 7, 2000.

[14] Defense Modeling and Simulation Office (DMSO). "High Level Architecture Rules, Version 1.3." Feb. 5, 1998. Online document at: http://www.dmso.mil/hla/tech/rules/.

[15] Defense Modeling and Simulation Office (DMSO). "High Level Architecture Interface Specification, Version 1.3." Feb. 5, 1998. Online document at: http://www.dmso.mil/hla/tech/ifspec/.

[16] Defense Modeling and Simulation Office (DMSO). "High Level Architecture Object Model Template, Version 1.3." Feb. 5, 1998. Online document at: http://www.dmso.mil/hla/tech/omtspec/.

[17] Defense Modeling and Simulation Office (DMSO). "High Level Architecture Federation Development and Execution Process (FEDEP) Model, Version 1.5", December 8, 1999.

[18] Mastaglio, Thomas W. "Enterprise Simulations: Theoretical Foundations and a Practical Perspective." *Proceedings of the 1999 Winter Simulation Conference*, Orlando, FL. December 1999.

[19] Pace, Dale K., "Simulation Conceptual Model Role in Determining Compatibility of Candidate Simulations for a HLA Federation" *Proceedings of The Spring 2001 Simulation Interoperability Workshop*, Orlando, FL. March 2001.

[20] Sarma, Debabrata. "DLLs for Beginners" Microsoft Developer Support, November 1996. Online Document at: http://microsoft.com/kb/default.asp/.

[21] CCRP, "NATO Code of Best Practice for C^2 Assessment Decisonmaker's Guide." 2002. Online Document at: http://www.dodccrp.org/nato_supp/pdf/NATO%20Decisionmaker_10_02.PDF.

[22] IEEE, "IEEE/EIA Industry Implementation of International Standard ISO/IEC 12207 : 1995 (ISO/IEC 12207) Standard for Information Technology Software life cycle processes" IEEE Std. No. 122070-1996, March 1998.

[23] Defense Modeling and Simulation Office (DMSO). "VV&A Recommended Practices Guide", Online Document at: http://vva.dmso.mil/.

[24] Law, Averill M., and Kelton, David W. "Simulation Modeling and Analysis," McGraw-Hill, Boston, MA. 2000.

[25] Lutz, Robert. "HLA Object Model Development: A Process View," *Proceedings of The Spring1997 Simulation Interoperability Workshop*, Orlando, FL. March 1997.

[26] Simulation Interoperability Standards Organization. Online at: http://www.sisostds.org.

[27] Craft, Mellisa, Poplar Mark A., Watts, David V., Willis, Will. "Network+ Exam Prep," The Coriolis Group, Scottsdale, AZ. 1999.

[28] Defense Modeling and Simulation Office (DMSO). "HLA RTI 1.3 Next Generation Implementation RTI-NG 1.3v3.2 Installation Guide", September 7, 2000.

[29] Defense Modeling and Simulation Office (DMSO). "High Level Architecture Federation Execution Details (FED) File Specification RTI 1.3 Version 3", July 31, 1998.

[30] Defense Modeling and Simulation Office (DMSO). "High Level Architecture Federation Security Process Version 1.2", February 16, 2001. Online document at: https://www.dmso.mil/public/library/projects/hla/guidelines/fsp1.2-final.pdf.

[31] Defense Modeling and Simulation Office (DMSO). "HLA Compliance Checklist Federate Version 1.3 (Draft)", May 4, 1998. Online document at: https://www.dmso.mil/public/library/projects/hla/compliance/chklst132d.pdf.

[32] Defense Modeling and Simulation Office (DMSO). "Modeling and Simulation Resource Repository (MSRR)", Online at: http://www.msrr.dmso.mil/.

[33] Prata, Stephen. "The Waite Group's C++ Primer Plus, Third Edition." Sams Publishing, Indianapolis, IN. 1998.

[34] Holzner, Steven. "Microsoft® Visual C++® 5 No experience required. ™," SYBEX Inc., San Francisco, CA., 1997.

[35] Defense Modeling and Simulation Office (DMSO). "Hands-On Practicum Programming with the Run Time Infrastructure." Alexandria, VA., Jun. 8, 1998.

[36] Systems Modeling Corporation, "*The SMART Files Library*", Sewickley, PA., 1997.

[37] Kelton, David W., Sadowski, Randall P., and Sadowski, Deborah A., "Simulation with Arena," McGraw-Hill, Boston, 1998.

[38] Defense Modeling and Simulation Office (DMSO). "High Level Architecture Interface Specification Annex B: C++ API" Online document at: https://www.dmso.mil/public/library/projects/hla/specifications/c_api_annex.pdf.

CURRICULUM VITA
for
Anton R. Lidums

DEGREES:

Bachelor of Science (Electrical Engineering), Old Dominion University, Norfolk, Virginia, May 2000

PROFFESIONAL CHRONOLOGY:
Northrop Grumman Newport News,
Newport News, Virginia
Senior Engineer, June 2001 – Present

Virginia Modeling Analysis and Simulation Center,
Suffolk, Virginia
Research Assistant, December 1999 – June 2001

Jefferson National Particle Accelerator
Newport News, VA
Co-op Technician, May 1999 – December 1999

Electric Motors & Contracting Co., Inc.,
Chesapeake, Virginia
Associate Engineer, May 1998 – May 1999

United States Navy,
Norfolk, Virginia
Nuclear Machinist Mate/Engineering Laboratory Technician, May 1991 – May 1997

SCIENTIFIC AND PROFESSIONAL SOCIETIES MEMBERSHIP:
Member, Simulation Interoperability and Standards Organization
Member, Golden Key National Honour Society
Member, Tau Beta Pi Engineering Honor Society
Member, Eta Kappa Nu Electrical Engineering Honor Society