LESSONS LEARNED FROM HUMAN-IN-THE-LOOP HLA IMPLEMENTATION

John P. Baker, Walter E. Bowen, and Michael A. Harris
Johns Hopkins University
Applied Physics Laboratory
Laurel, MD

ABSTRACT

A distributed simulation experiment conducted using a High Level Architecture (HLA) federation object model and the Runtime Infrastructure (RTI) was performed by the Johns Hopkins University Applied Physics Laboratory (JHU/APL). To explore the utility of the HLA and RTI for representative military problems, the experiment simulated two power projection scenarios. For the first scenario, a simulated Tomahawk guided missile with intelligent anti-armor submunitions was launched against an armor column. In the second scenario, an advanced hypersonic missile was simulated as an anti-TEL (Transporter Erector-Launcher) weapon following a Scud launch. In both scenarios, two Unmanned Aerial Vehicles (UAVs) were used to detect and track targets, one with Moving Target Indicator (MTI) radar and one with Electro-Optical and Infra-Red (EO/IR) sensors. This allowed initial targeting and, in the first scenario, dynamic re-targeting of the Tomahawk missile. The EO/IR UAV payload operator station (based on the Predator UAV) was simulated with virtual imagery in real-time and a user interface which allowed an operator to select and control the sensors dynamically. A Virtual Strike Battle Station (VSBS) with a virtual reality user interface was used to simulate the command and control center. Lastly, the Tactical Event System (TES) was simulated to provide detection information of Scud launch events in the second scenario.

This paper describes the HLA implementation and lessons learned for using the HLA for real-time human-in-the-loop simulations. Specific problem areas are described and a suggested "road-map" for building HLA/RTI simulations is presented.

ABOUT THE AUTHORS

Mr. John P. Baker is a Senior Engineer at the Johns Hopkins University Applied Physics Laboratory (JHU/APL) within the Electro-Optical Group of the Air Defense Systems Department. He has spent five years in the design of several fielded data acquisition systems including a recent successful demonstration of Unmanned Aerial Vehicular control system installed on a 688-class attack submarine. Mr. Baker holds Bachelor of Science degrees in Electrical Engineering and Computer Science from Washington University in St. Louis, a Master of Science in Electrical Engineering from Johns Hopkins University, and is pursuing a doctoral degree in Electrical Engineering at the University of Maryland College Park. He can be reached via e-mail at bakerjp1@jhuapl.edu.

Mr. Walter E. Bowen is a Senior Engineer JHU/APL. He has nine years of experience in modeling, analyzing, and developing concepts for weapon guidance, control, and navigation systems. His technical interests include fuzzy logic and object-oriented system analysis and design. Mr. Bowen received his BS and MS degrees in electrical engineering from the Virginia Polytechnic Institute and State University. He can be reached via e-mail at walt.bowen@jhuapl.edu.

Mr. Michael A. Harris is a Computer Scientist JHU/APL within the Strike Warfare Systems Engineering Group of the Power Projection Systems Department. Since joining APL, he has worked on a broad range of tasks including: distributed modeling and simulation, Air-Directed Surface-to-Air Missile (ADSAM) analysis, medical informatics, and virtual reality. He has BS degrees in Computer Science and Mathematics and an MA degree in Applied Mathematics from the University of Maryland College Park. He is currently pursuing a doctoral degree in Applied Mathematics from the University of Maryland College Park. He can be reached via e-mail at Michael.Harris@jhuapl.edu.

LESSONS LEARNED FROM HUMAN-IN-THE-LOOP HLA IMPLEMENTATION

John P. Baker, Walter E. Bowen, and Michael A. Harris
Johns Hopkins University
Applied Physics Laboratory
Laurel, MD

INTRODUCTION

The High Level Architecture (HLA) is being developed by the Defense Modeling and Simulation Office (DMSO), to facilitate interoperability among simulations and promote reuse of simulations and their components. The HLA has been designated by the Department of Defense (DoD) as the standard technical architecture for all DoD simulations.. The DoD has mandated that no further development occur after October 1, 1998 on any DoD-sponsored simulations that are not in the process of achieving HLA compliance. To achieve this mandate by 1999, the M&S community is developing expertise in using the HLA. This expertise is gained primarily through practical experience. By designing and building HLA-compliant federations that model real-world problems, skills and techniques are learned while studying the scenarios of interest. Out of these experiences arise lessons on how best to use HLA methodologies, and perhaps more importantly, how *not* to use them.

This paper describes an effort by the Johns Hopkins University Applied Physics Laboratory (JHU/APL) to use the HLA paradigm and associated Runtime Infrastructure (RTI) in a real-time simulation with humans-in-the-loop. In particular, this paper documents lessons learned for using the HLA. Properly applied, these lessons learned should expedite the development of future HLA-compliant simulations.

.

PROBLEM DOMAIN DESCRIPTION

To gain experience in applying the HLA to complex weapon systems analysis, two Naval interdiction concepts are being studied. Both concepts provide a means for forward-deployed US Naval forces to attack moving tactical targets deep inland.

Two Interdiction Concepts

The first of these concepts uses a modified cruise missile to attack a moving tank column. In contrast, the second concept uses a hypersonic missile to attack a Transporter Erector-Launcher (TEL) and its support vehicle after it has launched a Tactical Ballistic Missile (TBM.)

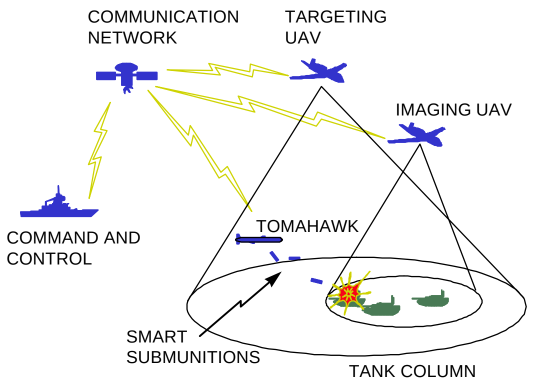

Figure 1: Armor Interdiction Concept

Both concepts integrate missile systems, submunitions, and a C^{4}ISR network into a single weapon system. The constituent components of these interdiction concepts are enumerated below.

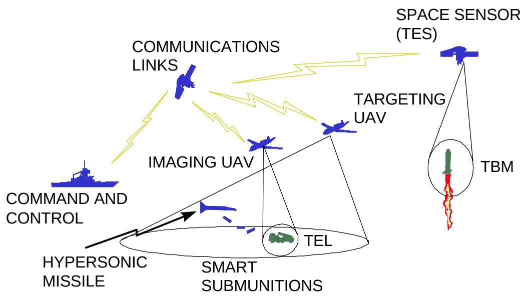

Figure 2: TEL Interdiction Concept

Sensors: Both concepts employ unmanned aerial vehicles (UAVs) to collect target location and imagery data that is transmitted in near-real time to a shipboard command and control center. One of these UAVs is equipped with a Ground Moving Target Indicator (GMTI) radar with a large sensor footprint that can provide targeting data on land vehicles distributed across the battlefield. The other UAV is equipped with high-resolution imaging sensors with a relatively small footprint that are used to provide positive target identification and to make battle damage assessments. Both UAVs are guided by human operators. The imaging sensor payload on the second UAV is also controlled by a human operator.

The TEL interdiction concept additionally employs the Tactical Event System (TES) to detect TBM launch events and estimate the location of the launcher.

Communication Network: A satellite communication network is used to provide connectivity between the Command and Control Center, the sensors (UAV and TES), and the missiles. Communication delays are modeled to simulate more accurately C^4I timeline issues.

Command and Control Center: The ship-based command and control center maintains a big-picture view of the land battle as it evolves. This information, along with knowledge of the weapon system, is used to make targeting decisions, missile launch decisions, and decisions to redirect in-flight missiles. This is done by human operators using computerized decision aids and other software that integrate down-linked missile and sensor data into a unified picture of the tactical situation.

An aspect unique to the armor interdiction concept is that the command and control center can re-direct the missile in-flight to a new terminal location if the intended target deviates from its predicted course.

Weapons: The armor interdiction concept employs an enhanced Tomahawk missile that can be redirected in-flight by the command and control center. The TEL interdiction concept employs the Quick-Reaction Deep-Strike (QRDS) hypersonic missile to attack the TEL and its support vehicles.

In both concepts, the missiles are armed with a payload of intelligent submunitions that are dispensed at the predicted target location. Because these submunitions can autonomously detect and fly to attack individual targets, a very large effective lethal footprint is obtained by the composite payload.

Weapon System Evaluation Criteria

The key question to be addressed with our system concept models is the following:

Does the integrated weapon system deliver its weapon with sufficient accuracy and timeliness for the lethal footprint to encompass the target?

To a great extent, the answer to this is determined by the ability of the system to predict the future location of a moving target. To be effective, the system must be able to detect and react to changes in the target's course. The overall responsiveness of this system is limited by those subsystem elements that have human operators interfacing with computer-aided display and machine control systems. In particular, the command and control system must incorporate the UAV data to synthesize a clear picture of the evolving tactical situation. This allows the human operator to make informed judgments about the situation and react accordingly. For this reason, our study focuses on the man-machine interface elements of the weapon system.

HLA REPRESENTATION OF THE PROBLEM DOMAIN

Federation Object Model (FOM)

The Federation Object Model (FOM) defines the federation hierarchy of object classes and shows how these object classes interact. This section describes the object class hierarchy of the federation used to represent the problem domain described above. (See Figure 3) The object classes on the right side of Figure 3 are concrete object classes that are instantiated as objects in member federates of the federation execution. The Vehicle, Surface Vehicle, and Air Vehicle classes are the super classes used to derive the concrete classes. The Vehicle class contains the physical position and orientation data required to relate one object to another. The hierarchy, along with the class interactions, was documented using the Object Model Development Tool (OMDT).

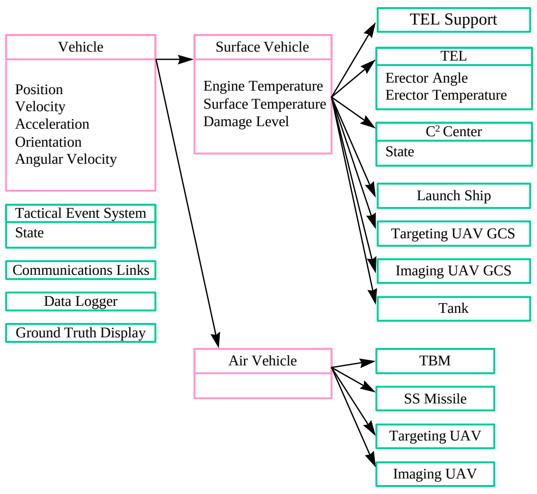

Figure 3: Federation Object Classes

The FOM was the backbone of the simulation effort. The participants hammered out an initial version of the FOM which was used to coordinate activities throughout the project. We believe that a good initial FOM is an essential tool to build an integrated, interactive simulation.

Description of Participating Federates

This federation includes a wide range of federates. Two of the federates contain real-time user interfaces which allow human-in-the-loop interactions: the Virtual Strike Battle Station (VSBS) and the Imaging UAV GCS. The other federates are purely analytic models. The federates in this simulation are described below.

Virtual Strike Battle Station: The Virtual Strike Battle Station uses virtual reality tools to provide an immersive environment for a futuristic strike battle station within the setting of the Command Information Center (CIC) of a DDG-51 class Navy ship. The immersive setting allows a user to walk through the CIC and sit at the Virtual Strike Battle Station (VSBS). The VSBS is a human-in-the-loop federate that allows an operator to view and interact with entities in the HLA simulation. VSBS consists of three main components: the virtual reality interface, the RTI communications process, and the Strike Manager process. The virtual reality interface is responsible for communicating visual information to the user via dynamically updated hanging displays viewed through a head-mounted display. The user interacts with objects in the exercise through a data glove and voice recognition interface. A 3-D GeoView is also being developed which will allow a user to view and interact with HLA objects displayed in three-dimensional space over a virtual table top. The RTI communications process handles all of the low-level RTI communications tasks such as sending and receiving interactions, processing entity attribute updates, and RTI time management. The Strike Manager process handles all of the computational and coordination functions of the VSBS. The virtual reality interface, the HLA communications process, and the Strike Manager process all communicate via TCP/IP sockets.

During the simulation, the VSBS operator is responsible for receiving sighting and track reports from the imaging and targeting UAVs as well as satellite-based Tactical Event System launch queues. The operator can act on this information by issuing commands such as launch orders, imaging UAV tasking orders, and Tomahawk redirection tasking orders. The commands, initiated by voice or gesture, are sent as RTI interactions to the appropriate receiving entities via the RTI communications process.

Imaging UAV GCS: The Imaging UAV GCS (Ground Control Station) federate is modeled after a Predator UAV. The electro-optic and infrared (EO/IR) sensors are modeled and the flight dynamics are consistent with the operating envelope of the Predator. The Synthetic Aperture Radar (SAR) sensors were not modeled. The Imaging UAV is used for target identification and Battle Damage Assessment (BDA).

A simulated payload operator's station, designated the Ground Control Station (GCS), presents an operator with a synthetic image of what a UAV would see flying over a real-world analog of the virtual battle-space. Updates of vehicle locations and speeds are used to dynamically render simulated objects on the operator's screen. The operator also has the capability to dynamically pan, zoom, and switch sensors much as a real payload operator would. The UAV sensor view cones are accurately rendered to provide the same "pinhole" viewing areas found in real UAV sensors.

The graphical visualization is performed by third-party software which accepts Digital Terrain Elevation Data (DTED), representative overhead terrain imagery, models of imaged targets, and dynamic position and orientation inputs. It renders the resulting dynamic scene in real-time. The dynamic data reflecting the positions of the UAV and the target objects is generated from an HLA federate. This federate receives updated object attributes and passes them to the visualization software via a UNIX socket-based connection.

The Imaging UAV GCS is designed to resemble a payload operator station for a Predator UAV. Such a station allows a payload operator to monitor the sensor data (video) while controlling the look angles, zoom, and sensor in use. The visualization system harnesses a high-end Silicon Graphics workstation to render realistic, color landscapes and targets as they would appear through a UAV sensor package. As the operator adjusts the pan, tilt, and zoom, the viewing frustum is altered in real-time. In addition, a separate federate propagates the flight-model of the UAV which periodically (via the RTI) updates the viewpoint of the UAV. (See Figure 4)

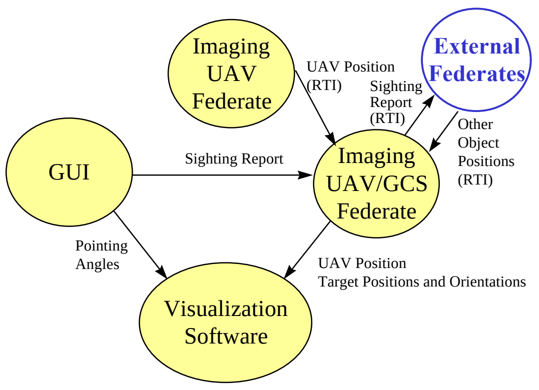

Figure 4: Imaging UAV GCS Software Architecture

Digital Terrain Elevation Data (DTED) files are overlaid with representative overhead imagery of the appropriate terrain type to generate landscape imagery. OpenFlight models provide the physical properties of the target vehicles and real-time RTI attribute updates are incorporated and smoothed to animate the scene.

IR imagery is currently mimicked by altering the color tables to show a gray-scale scene rather than a color scene. Some target object temperature attributes are passed to the GCS via the RTI and are used to modify the brightness of various features in the object. Thus, a tank that has been running for a time is brighter than one that has been inactive. The IR modeling at this point is very low fidelity and does not incorporate radiometric techniques.

The visualization software was written using the Performer libraries supplied by Silicon Graphics, Inc. Performer was chosen due to its wide availability, flexibility, and power. As RTI updates of changes in the position of the UAV and external federates appear at the GCS Federate, the relevant data are passed to the separate Performer process via inter-process communication protocols. The Performer application then applies these new values to its current "world state" and renders the resulting view. The Performer application renders data slightly delayed from real time so that data interpolation can be used to produce smooth animation.

Analysis and Support Federates

Imaging UAV: The Imaging UAV federate supports the Imaging UAV GCS by supplying a flight model which outputs realistic position updates based on the Predator vehicle's flight envelope. This UAV 'pilot' is a software program that receives tasking orders from the C^2 center and guides a flight model to a loiter point and performs a loiter pattern over the designated area. This program is implemented as a separate federate from the GCS federate. The GCS federate then subscribes to the UAV pilot federate to establish the eye-point for the visualization software.

Targeting UAV and Tomahawk Missile: The targeting UAV and the Tomahawk missile are modeled in the same federate. Both models use a common six degrees-of-freedom bank-to-turn flight kinematics model. Aerodynamic derivatives, linearized about a single trim condition, are used to model the aerodynamic forces on the vehicle. In addition, both models use a GPS inertial navigation model, and a mission waypoint guidance algorithm. The UAV includes a wide-area GMTI radar model, and the Tomahawk includes a submunition payload model.

Launch Ship: The Launch Ship is modeled as a generic Surface Vehicle. In the scenarios the ship is stationary but capable of motion. As currently implemented, the Launch Ship publishes its location and provides launch point coordinates for the Tomahawk and QRDS missiles.

Quick Response Deep Strike Missile: The QRDS missile cruises from the Launch Ship towards the TEL target at hypersonic speed. A dive, pull out, deploy maneuver is executed to dispense a payload of multiple Brilliant Anti-Tank (BAT) submunitions. The QRDS object publishes the usual motion attributes. It interacts with the TEL and support vehicle to compute their Damage Level attributes using a simple footprint and probability-of-kill algorithm. A model computes the complete trajectory of the QRDS at launch time using the Launch Ship and TEL locations.

Tactical Event System: The Tactical Event System (TES) simulates the ALERT software which processes data from space-based sensors that can detect a tactical ballistic missile (TBM) in flight. The state vector and launch point are estimated from the estimated TBM exhaust plume temperature. The launch point estimate is then passed to the command and control center.

Communications Link: The purpose of the Communications Link federate is to provide representative delays associated with sending data across real-world tactical communications networks or tactical data links (TDLs). A small number of TDLs appropriate for the scenarios were identified, and delays were modeled stochastically from probability distributions that adequately reflect the latency time typical for those TDLs. All HLA interactions which represent tactical communications are sent to the Communications Link federate. This federate determines a delay time and relays the message (as another interaction) to the intended recipient after that time has elapsed.

Tactical Ballistic Missile: The TBM has a boost phase (with an exhaust plume) and a ballistic phase. The dynamics of the boost phase of the TBM trajectory are published to the federation using a lookup table produced by a higher fidelity model. The missile also publishes an Intensity attribute to the federation that changes when the SCUD enters its ballistic phase.

TEL and Support Vehicle: A TEL (Transporter Erector-Launcher) and an associated support vehicle are modeled as generic Surface Vehicles. The vehicles publish position, motion and temperature data to the federation. The stationary TEL launches a SCUD missile and begins moving to a new location accompanied by the support vehicle. The TEL Erector Angle and Erector Temperature are also published. The erector gradually cools after the SCUD launch. This is modeled using an exponential curve. After munitions are deployed against the TEL and support vehicle, a Damage Level attribute is published.

Tank Column: A tank column is simulated as a group of tanks (Surface Vehicles) moving along a grid of roads. Speed and configuration of the column are modeled based on realistic tactics. Tanks also publish temperature attributes that influence detectability by various sensors. The tank column performs turning maneuvers during the simulation while the Tomahawk missile is in the air. After munitions are deployed against the column, individual tanks publish a Damage Level attribute.

Data Logger: The logging of data produced during an HLA federation execution is far more complicated than it was for Distributed Interactive Simulation (DIS). Logging and analyzing HLA federation execution data has been the topic of much debate and many papers within the M&S community. For DIS, it was sufficient (more or less) to simply "catch" all of the data packets, or Protocol Data Units (PDUs), and store them in a file. For an HLA federation execution, on the other hand, it is not efficient to have one (logger) federate receive all data passed around during the exercise. This approach is not effective for many reasons, including the fact that it counteracts the advantages in the reduction of network bandwidth consumption provided by the HLA declaration management and data distribution service categories.

For the purposes of this experiment, however, a "simple-minded" DIS-like logger captures all object discoveries, attribute updates, interactions, and object removals and logs them. This is sufficient to provide the post-execution debugging, analysis, and playback support required by the participants. Such a log and playback utility has been created which leverages the run-time type identification features of the RTI to achieve "FOM-independence."

Ground Truth Display: In order to provide visual representation of the scenario as it unfolds, an interface between the HLA RTI and an existing 3D graphics system, FTEWA, was established. FTEWA, or Force Threat Evaluation and Weapons Assignment, is a prototype air-defense shipboard system developed at JHU/APL which is currently installed onboard the USS Kitty Hawk, the USS Cowpens, and the USS La Salle. FTEWA provides a real-time 3D picture of the tactical situation, and a number of planning, analysis, and alert tools to assist the commander in making informed decisions in time-critical situations. The graphics capability of FTEWA is used to display the positions and orientations of all entities in the scenario, with accurate 3D models, as the federation execution progresses. The models used to represent scenario entities are dynamically scaled to enable effective engagement-level and theater-level views of the scenario.

Network Architecture

Each of the federates in the exercise was hosted in a secure facility within JHU/APL. The facilities are the Command and Control Systems Laboratory (CCSL), the Combat Systems Evaluation Laboratory (CSEL), the Mission Planning Development Laboratory (MPDL), the Space Department War Room also known as the Defense Systems Analysis Center (DSAC), and the Warfare Analysis Laboratory (WAL). These classified facilities are connected by a secure fiber optic local area network. The federates are hosted on a variety of computer platforms as indicated below: (See Figure 5)

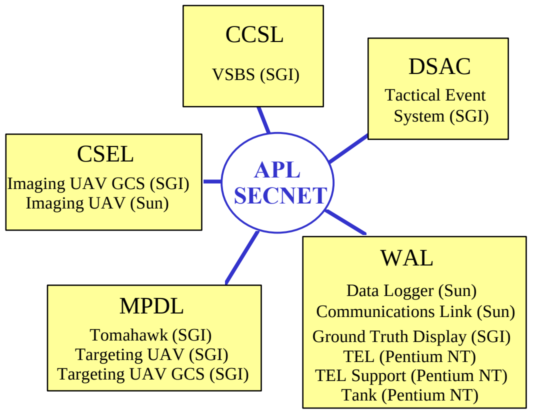

Figure 5: Secure Network Architecture

LESSONS LEARNED

An old saying applies: How do you get good judgment? Experience. How do you get experience? Bad judgment. In the course of creating this HLA federation, many problems were discovered and overcome. Some of the solutions and methodologies employed have wider application than this single instance. Future projects may benefit from our experiences in the areas of FOM creation, coding for the RTI, and human interfaces under HLA.

FOM Creation Process

This section describes the process used to create the FOM for the HLA federation described in this paper. While we found the FOM to be a very useful “blueprint” for coordinating efforts among the federates, it alone was not sufficient. Thus, we found it necessary to supplement the FOM with additional information.

The development of the Federation Object Model (FOM) was a three-step group-oriented process. The first step was the dissemination of a high-level description of the domain problem among the federation analysts. The second step was to develop consensus on the federation structure. The third, and final step, was writing the FOM. This three-step process is described below. See Reference [1].

Step 1: Disseminating a high-level description of the domain problem

As is true of all object-oriented simulation designs, a high-level understanding of the domain problem is needed prior to creating an object model. Thus, scenario documentation, while not explicitly enumerated as part of the Object Model Template (OMT), is a prerequisite to developing the FOM. The participating analysts, selected for their domain knowledge in specific subsystems, were, for the most part, unfamiliar with the larger weapon system concepts. To correct this deficit, a document describing the weapon system concepts and proposed test scenarios was written and disseminated among the analysts.

Step 2: Mapping the domain problem into an HLA federation object structure

After the analysis participants reviewed the weapon system concept descriptions, an all-day meeting was held among the simulation analysts. Specifically, the objectives of this meeting were to specify the object structure of the HLA federation, to parse this federation into constituent federates, and to assign responsibilities for each of the federates. This meeting was led by an individual who worked at a blackboard gathering inputs from the subsystem analysts.

Listing the object classes: During the first part of this meeting, we created a list of the concrete objects and corresponding classes that constituted the domain problem. The primary purpose of this list was to completely represent the problem domain. For this reason, we restricted our list to concrete classes, and deferred discussion of class hierarchies to later. Having listed the constituent concrete classes and class objects in the federation, object modeling responsibilities were assigned among the analysts.

Tabulating the object class attributes: Having compiled a complete list of classes and class objects, the next step was to create the FOM class attribute table.

The first step toward creating the FOM class attribute table was to determine how the object ownership was to be distributed among the constituent federates. This is needed because the FOM class attribute table should only list those class attributes that are passed <u>through the RTI from federate to federate</u>. The only way to identify these attributes is to know which object is in which federate. Note that our federation did not invoke any of the RTI services that transfer object ownership on request between federates.

Although it is not part of the FOM, we found it necessary to select an earth model and coordinate system. Without an established agreement on this, it would be impossible to avoid ambiguities in interpreting much of the exchanged attribute data. We selected a WGS84 earth model and expressed all translational and rotational attribute data with respect to the Earth-Centered-Earth-Fixed (ECEF) system. All federates linked to a standard library of subroutines to translate between the ECEF and Latitude, Longitude, Altitude coordinate systems.

We used a systematic approach to create a class attribute table that only contained necessary and sufficient attributes. First, we stepped through each listed object and had the modeler of that object list those attributes that he thought would be needed by an object in another federate. After this was done, we went back through each of the tabulated attributes and asked, “Does any object in any other federate need to know about this attribute?” When the response was negative, the attribute was struck from the list. In addition, we asked if there were any class attributes that were not currently listed, but were needed by other federates and made attribute additions accordingly.

Alternatively, we could have skipped asking the object modeler for attributes that he thought might be useful for objects in other federates. However, we found this to be a useful way to exchange modeling information.

Tabulating the interactions and interaction parameters: In the latter part of the meeting, we tabulated the FOM object interactions and interaction parameters. This interaction table was created by stepping through the federates, as was done with the object class attributes.

Initially, there was some confusion about class interactions that arose from the fact that object class interactions are used differently than C++ member functions. In particular, there are two salient differences. First, interactions send information “one way only.” Thus, it takes a query interaction and a response interaction to approximate a function call that returns data. Second, unlike C++ class member functions, RTI-supported interactions are directed to classes, not class objects. To effectively direct an interaction to a single object, one must include the identity of the receiving object as an interaction parameter. The interaction will be sent to all the instantiated class objects, each of which must check the parameter list to determine if it is the intended recipient.

Step 3: Documenting FOM Meeting Outputs in OMT Format

The third step was to assign a FOM point-of-contact tasked with compiling the tabulated data from step 2 into an OMT document and a FED file using the OMDT tool developed by AEgis Research. This point-of-contact was also tasked to organize the concrete classes into an inheritance hierarchy. These FOM and FED file documents were maintained by the point-of-contact on a homepage on the JHU/APL intranet. Corrections and modification to the FOM were made by e-mail to the FOM point-of-contact. No further "all-hands" FOM development meetings were needed, as we found feedback via e-mail to be sufficient.

Coding Techniques

Since the federate developers did not have the benefit of an existing base of reusable code, each federate was required to create all of the low-level HLA/RTI federate ambassador code from scratch. Collectively the model developers found the coding of the federate ambassador segments to be tedious and repetitive. There was a large amount of duplicated effort because each developer created RTI ambassador code segments to perform similar functions but implemented these segments with slightly different approaches. This created more work later in the testing phase where similar bugs were discovered and repaired multiple times because some of the approaches had similar problems. Although our HLA/RTI development effort was not optimal in terms of time efficiency, we did succeed in increasing corporate knowledge of the RTI/HLA low-level communications procedures. Attaining this corporate knowledge was one of the objectives of our project, so the extra time invested was not wasted effort. Below, we suggest some strategies for reducing the time and staffing requirements for HLA/RTI development.

One way that we might have reduced our federate integration costs is to have one developer create a generic federate ambassador template. The template would include all of the interaction and attribute handling code that is needed by all of the federation members. Federation members could then use this template for their individual federate ambassador development. Methods that are not required by a given federation member could simply be cut from that member's copy of the template. This approach would greatly reduce the amount of duplicated effort in both the coding and testing phases of the federate ambassador development. In addition, it would guarantee a consistent data-type interpretation of exchanged data.

Another improvement to our federate ambassador coding would be to represent the interactions as object classes. If each federate had a local copy of a set of commonly used interaction object classes, they could use the class-provided interaction services rather than having to custom-code them into every ambassador. By promoting code reuse, these interaction object classes could potentially yield a significant reduction in the cost of federate ambassador coding. For example, in our scenario the Tomahawk and QRDS missile send target kill interactions indicating that a target in the simulation has been killed. The tank, TEL, and TEL support federates all receive the target kill interaction and remove themselves from the simulation if they are identified as killed. If the target-kill interaction is coded as an object, each object that uses it can call the appropriate send or receive member functions without creating new code in each federate ambassador. (See Figure 6)

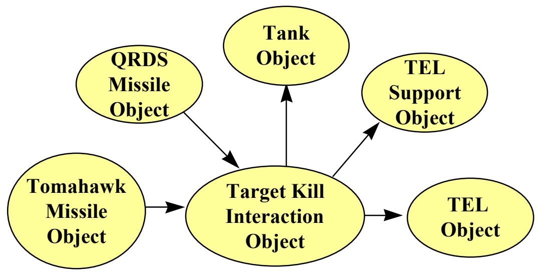

Figure 6: Object Oriented Code Structure

The interaction objects would encapsulate all of the data needed to send and receive interactions to the RTI. This strategy reduces the amount of redundant code and increases the number of data-type mismatch errors that are caught during compilation avoiding costly runtime debugging.

Finally, there is a need for tools such as an automatic code generator to create the federate ambassador for each federate. Currently, coding for the HLA requires a great deal of time and tedium to implement simple functions which could well be automated (e.g., extracting data from an attribute or parameter list). Such a code generator needs to take as input a file that contains all of the information contained in the federate ambassadors. (See Figure 7) The OMDT- developed by the AEgis Research Corporation outputs a file (the OMDT file) that would be suitable for such a code generator. This file contains data type information and information indicating the sending and receiving objects for interactions. The source code generated would provide the template code for each HLA object. "Stub" member functions would be created which would allow programmers to quickly customize the generic template to each specific model.

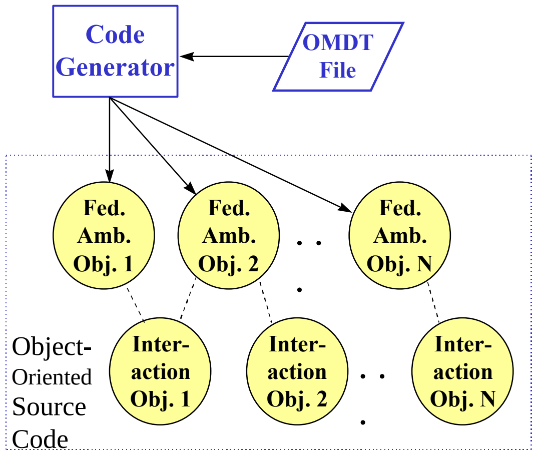

Figure 7: Automated HLA Code Generation System

There is also a need for a common tool set for federation developers. This tool set could provide facilities for coordinate conversions, external object discovery and tracking, and attribute update regulation. For example, as a result of our federation development we had several different implementations of methods for discovering and tracking objects external to a given object in the exercise. This redundancy could have been eliminated if such a tool kit had been available. Integrators of legacy and other types of systems who do not want to use the tool kit code would not be required to do so. On the other hand, developers of new models could use common routines for tasks like coordinate conversion to produce more accurate results. For example, in our federation we shared a common set of coordinate conversion routines.

Human Interface Issues

The Virtual Strike Battle Station (VSBS) and the Imaging UAV Ground Control Station (GCS) are the two human-in-the-loop federates in our exercise. The architecture of both federates is similar in that the HLA/RTI functionality is separated from the human interface. This separation allows the human interface to operate somewhat independently from the RTI. For example, if either of the federates were required to operate in a DIS exercise there would be only minor changes in the human interface modules. The HLA/RTI process would simply be replaced by a DIS communications process.

Virtual Strike Battle Station: The VSBS human interface (VSBS-HI) consists of a VR4 CiberMaxx head-mounted display, a Sense 8 pinch data glove, and an Ascension Flock of Birds magnetic position tracking system. These components together with the Sense 8 WorldToolKit simulation engine hosted on an SGI Reality Engine are the core of the user interface. Voice recognition is accomplished using the Kersweil Voice software package on a Pentium-based PC. The VSBS-HI receives images of the Strike Manager console from the Strike Manager process. These images are incorporated into the virtual environment as hanging two-dimensional displays. The displays are updated regularly to provide situational awareness as the exercise unfolds. The user initiates actions by gesturing with the data glove or speaking voice commands. The gesture and voice recognition interfaces translate these actions into messages that are sent to the RTI and Strike Manager processes for interpretation and processing.

The VSBS-HI constrains the federation to execute in real time because the human operator naturally perceives and acts on data in real time. Similarly, the Imaging UAV Ground Control Station (GCS) requires real-time federation execution. In addition, the GCS places a much higher data-rate requirement on the federation members because it requires the operator to view, interpret, and act on real-time video. User interaction with the GCS is achieved through the Interactive Visualization Interface.

Interactive Visualization Interface: The Imaging Unmanned Aerial Vehicle (UAV) Ground Control Station (GCS) emulates a Predator UAV payload operator station. The payload operator in the real-world system controls the sensor, pointing angles, and zoom factor of the sensor package. The operator requires immediate feedback through the video generation system when a control is activated. For example, when the operator presses the "pan" button, the display should pan immediately. Real-world delays are an exception and should be modeled accurately.

More to the point of a distributed interactive simulation, a surface object under surveillance should move smoothly and realistically over terrain. It must not be allowed to "fly," "tunnel," or "teleport." Given the fact that some simulations will not generate updates at the required frame rate for smooth animation (approximately twenty-four frames per second), an interpolation or extrapolation technique is required. (See Reference [2]) By slightly delaying the rendering of data, linear interpolation may be used instead of extrapolation, thereby removing issues associated with inaccurate extrapolation of future events. This delay must be small enough (e.g., 100 milliseconds) so that there is no performance bias introduced into the simulation. In addition, the extrapolated positions must be adjusted further to constrain the objects to the terrain altitude.

Human Interface Lessons Learned: Incorporating a human interface into an HLA-compliant distributed simulation presented several challenges. While not as restrictive as a hardware-in-the-loop simulation, real time performance issues had to be addressed.

One of the design rules enforced for HLA-compliant simulations with complex user interfaces was the concept of separation between the user interface and the HLA/RTI interface. Separate processes and/or threads should be used to allow smooth updates of the user interface data without being tied to the HLA/RTI “tick” loop. This allows the user interface to update at an independent rate from the HLA/RTI “tick” loop. Without this separation, small delays in the RTI or other federates can cause jitter in visualization renderings. Sudden jumps in object positions can be filtered or interpolated to produce smoothly moving imagery. (See Reference [2])

CONCLUSION

Summary

This HLA-compliant simulation addressed several issues involved with use of the RTI in a real-time, human-in-the-loop situation. A hierarchical federation containing fifteen (15) instantiated classes was used to investigate two time-critical power-projection scenarios. The simulation was distributed over several JHU/APL laboratory facilities connected by a secure fiber optic network. Real-time visualization hardware and virtual reality workstations provided the human-in-the-loop elements (UAV Payload Operator and Command and Control Operators) to the simulation. Several lessons learned are presented and suggestions for future HLA/RTI tools are made as follows:

- -- Establish a scenario and FOM early in the development process
- -- Use the FOM as a coordination mechanism throughout the development process
- -- Establish a generic federation ambassador template for use by all federates
- -- Utilize object-oriented coding methodologies to implement federate ambassador
- -- Investigate possibility of automated code generation techniques for federate ambassador template
- -- Separate user interface processes from HLA/RTI interface processes

Future Work

Future related efforts could include the following:

- -- Incorporating a Synthetic Aperture Radar model into the Imaging UAV simulation
- -- Expanding the threat simulations to improve the realism of the threat behavior
- -- Formalize the FOM for inclusion in the Modeling and Simulation Resource Repository (MSRR)

REFERENCES

[1] Lutz, Robert, "HLA Object Model Development: A Process View," Presented at Simulation Interoperability Workshop, March 3-7, 1997

[2] Lin, Kuo-Chi, et. al., "Smoothing of Dead Reckoning Image in Distributed Interact Simulation," J. Aircraft, Vol. 33, No. 2, 1995, pp. 450-452