Development of a Surface-to-Air Missile T&E Federation

Jason L. Lucas
TYBRIN Corporation

Robert A. Kent
96 CG/SCTOS

ABSTRACT

This paper focuses on the development of the Surface-to-Air Missile Federation (SAM Fed), a Test and Evaluation (T&E) High Level Architecture (HLA) federation. The SAM Fed evolved from a legacy constructive real time/post mission hardware-in-the-loop (HITL) Surface-to-Air Missile (SAM) simulation. The process to transition from a stand-alone simulation to a distributed HLA federation as well as the problems experienced and lessons learned will be presented. This paper will also discuss the merit of standardizing internal federate interfaces to facilitate reuse.

KEY WORDS

Surface-to-Air Missile, SAM Federation, Test &Evaluation, HLA, RTI

INTRODUCTION

It is often difficult to break deterministic repeatable test and evaluation simulations into a distributed simulation regime. This is especially true for closed-loop hardware-in-the-loop (HITL) simulations that rely on very high data rates. However, when an open-loop testing scenario exists there are several options available to distribute the simulation to allow for active and passive participation in the testing. This paper presents an approach to migrate multiple stand-alone surface-to-air missile (SAM) simulations into distributed operation by utilizing High Level Architecture (HLA).

BACKGROUND

The 96th Communications Group, Modeling and Simulation Support Flight (96 CG/SCTO), Eglin AFB, FL, is responsible for assessing and introducing emerging modeling and simulation (M&S) capabilities to its customers for munitions and Command and Control (C2) systems testing. To this end, in cooperation with

AFDTC/DRC and funding support from the Central Test and Evaluation Investment Program (CTEIP) Foundation Initiative 2010, a Development Test Cell (DTC) has been established in the Freeman Computer Sciences Center at Eglin Air Force Base to support the migration to HLA. This DTC provides the resources for development and testing of HLA applications at Eglin AFB.

Eglin began developing the HLA DTC in August 1997. By November, the equipment was procured, people were trained, and an initial HLA demo was operational. The initial efforts were aimed at understanding the requirements of HLA and assessing the impact of making a legacy model set HLA compliant.

LEGACY MODEL SET

The legacy model set consists of high fidelity real time SAM and antiaircraft artillery (AAA) flyout models which provide miss distance as a measure of effectiveness in support of Electronic Combat testing at several DoD test facilities. These flyout models were developed under a range improvement initiative funded in 1994. They are kept under strict configuration control in the Flyout Model Information Center (FMIC) by the Modeling and Simulation Support Flight.

Many of the FMIC models have been independently validated against approved baselines. The remaining models will be validated when funding is available. To date, the FMIC contains fourteen missile simulations, two gun simulations, and off-line executives used for post mission data reduction. All of the software developed for the FMIC adheres to sound software engineering methods, including comprehensive documentation. Since the models are used to support testing at multiple facilities, we developed a set of interface standards known as the *Standard Interface for Missile Simulations* (SIMS) and the *Standard Interface for Gun Simulations* (SIGS) which address model taxonomy, model interface, coordinate systems, and sign conventions. The models are all written in ANSI standard FORTRAN that allows the models to be compiled and run on a wide range of computer platforms.

LEGACY SYSTEM ARCHITECTURE

The FMIC SAM simulations predominately model the missile launch computer, guidance computer, autopilot, and airframe systems. To simulate a command guided missile engagement, the SAM flyout model must be run simultaneously with a target model and a target tracking radar (TTR) model.

The order of operation for the engagement scenario begins with the target model computing the aircraft position, velocity, acceleration, and orientation. Next, the TTR reads the target position and velocity and computes the azimuth and elevation angles and

range of the target with respect to the TTR position. Finally, the SAM model reads the TTR data, computes launch angles, guidance commands, fin deflections, airframe aerodynamics and updates the missile position, velocity, accelerations, and other missile state information. In addition, the true target position is read by the missile model to compute the point of closest approach, a.k.a. miss distance.

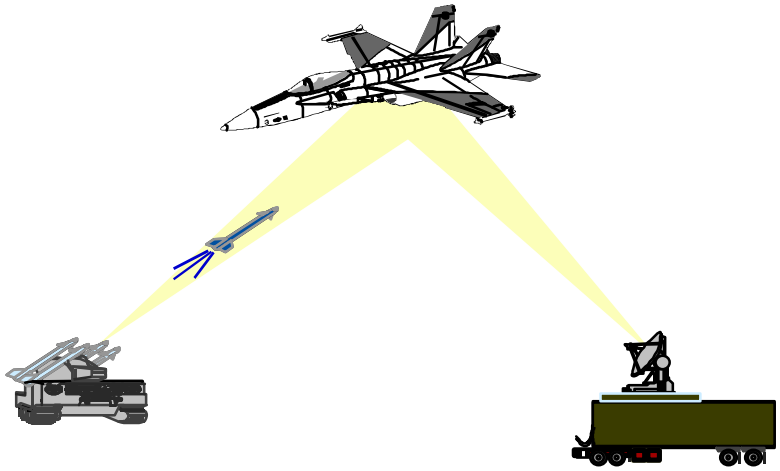

Figure 1. SAM Engagement Scenario Participants

A frequently used in-house FMIC model evaluation and demonstration executive program named PCDEMO generates the target position and velocity and emulates the TTR functions by computing angles, angle rates, range, and range rates. The executive sequentially calls the target, TTR, and missile model modules ensuring the modules are in phase with each other. PCDEMO then steps forward in time at the specified integration time step for the simulation. The integration time step for each of the legacy missile models is unique and is a multiple of the update rate of the TTR. The PCDEMO executive serves as the basis for this HLA SAM federation.

OBJECTIVE

Due to the progression of HLA to the forefront of distributed simulation and the promise of its enhanced performance over DIS, a SAM Federation (SAM Fed) was conceptualized to leverage from a validated legacy model set and transition into the distributed regime. The key constraint for the SAM Fed was to perfectly duplicate the operation of the legacy SAM simulations. The primary benefit of the SAM Fed would be the ability to “plug and play” missile models into a missile federate without any modifications to the missile model code. Such a plug and play ability would be possible due to the SIMS interface. Hence the missile federate would function as a generic HLA interface. A further benefit of the SAM Fed would be to allow validated SAM systems to be reused in both the testing and training communities.

SAM FED ARCHITECTURE

In an object centric view of the command guided engagement SAM scenario, the participants are a target object, a TTR object and at least one missile object. It could be argued that the TTR should not be an object since it produces information about the target object. However, in our testing scenario the TTR is co-located at the SAM launch site, and can be deceived to provide false information about the target. We chose to allow the TTR to be declared an object (owning its own attributes) because of its ability to always provide antenna pedestal angles regardless of whether a target was present. A controller federate was conceptualized to perform the simulation management functions. The controller also can function as a conduit for non-HLA surrogate visualization tools.

DEVELOPMENT

Our development efforts are presented here in the order we actually performed them. We deviated from the DMSO Federation Development Process (FEDEP) significantly during the prototyping stages, but later used portions of the FEDEP for the formal SAM Fed development[2].

It is also important to note that the prototype development efforts were pursued with a somewhat minimalist mindset. Many of the Federation Object Models (FOMs) and Simulation Object Models (SOMs) we reviewed seem designed to address the needs of many federates, and could prove cumbersome at very high data rates. We hoped to implement the HLA between the SAM Fed participants in a manner where extremely high data rates would not be encountered, and the volume of data transported would be relatively small.

The SAM Fed is required to operate with synchronous updates for each time step and with sequential stepping from the target, to the radar, and then to the missile(s).

HLA EXPERIMENT

In order to understand the HLA and the behavior of the Run Time Infrastructure (RTI) version 1.0.3, a feasibility experiment was initiated to implement a working SAM prototype federation by radically modifying and building upon the DMSO Hello World federate tutorial[3].

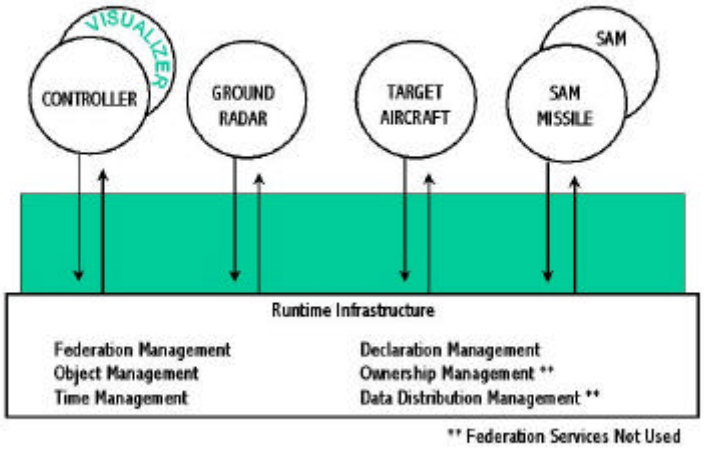

Figure 2. SAM Federation

The SAM prototype federation is made up of two SAM federates, a target federate, a radar federate, and a controller federate complete with a visualizer. Figure 2 shows the logical relationship between the federates and the RTI. The PCDEMO executive program was chosen to serve as a framework for the new federates in the SAM prototype Fed. In preparation for building the SAM Fed prototype, we converted the PCDEMO code from FORTRAN to C. An unclassified generic missile model was chosen as the SAM for use in the experiment. Because the legacy model set and the generic SAM each use the SIMS interface, conversion to the HLA follows the same implementation for all models in the set. The command guided engagement scenario selected utilizes a target profile to evaluate missile performance. The target portion of the PCDEMO code was isolated to form the framework of what would become the target federate. The PCDEMO radar section was isolated to form the framework for what would become the TTR federate. The PCDEMO missile model portion was isolated and duplicated to form the framework for what would become the two missile federates that we call the generic SAM modules.

SUBCOMPONENT DESCRIPTIONS

The target code generates a target track from a scripted profile that is provided interactively at runtime by PCDEMO, but we hard coded it into the target federate. The target profile flies a point target in level flight through two inbound course changes, crosses by the SAM launch site, and changes course once while outbound. The radar code computes radar tracking parameters from the target data. The radar was purposely kept as a perfect tracking system, not producing off-boresight or monopulse errors, to emulate the radar operation from PCDEMO. The missile model framework monitors missile control flags to determine when it is appropriate to call the missile modules for launch and guided flight, or to terminate flight. The Hello World tutorial is a simple HLA federate that simulates an increase in population over time. It registers one instance of the

HLA object class "Country" and updates its name and population attributes as they change. It is important to note that in the RTI distribution, Hello World does not incorporate time synchronization. Beginning with Hello World, the SAM prototype Federation was developed in three phases.

TIME MANAGEMENT

In the first phase of development, we addressed time management. We modified the "HelloWorld.fed" file to specify FED_RELIABLE transportation and FED_TIMESTAMP ordering for the "Country" class attributes. We then modified the Hello World federate ambassador to set the RTI Support Service setTimeConstrained to "True". This change defined that events only become eligible for presentation to a federate when it can be guaranteed that no time stamp ordered (TSO) events with a lower time stamp will be received. This ordering only applies to events that are designated by the sender as being TSO. Additionally, we modified the ambassador to call the turnRegulationOnNow RTI Support Service and set the Time Management Service setLookahead to 1. Upon execution of the Hello World federation, each Hello World federate that joined, published its data, and subscribed to the other federates' data in the same order, thus providing sequential, synchronous operation required by the SAM Fed.

SOM CONVERSION

In the second phase of prototype development, we converted the Hello World SOM "Country" class "population" attribute to a complex data type. The complex type consisted of a data structure of 100 doubles. The resulting type allowed for an assessment of the "loading" capability of the RTI.

MERGING SAM OBJECTS WITH CONVERTED SOM CODE

In the third phase, we created an instance of the converted SOM Hello World code corresponding to each object in the SAM Federation and a controller. In each instance, we replaced the population growth by the appropriate calls to the legacy code modules. We set the "Country" class "Name" attribute to identify the SAM object instantiated by the federate. Next, we modified each of the new federates to map the SIMS variables needed by each of them to their "Country" class "Population" attribute. The variables that were mapped to the complex type were chosen based on the minimum information exchange needed between the federates. Each of the new SAM prototype federates published to and subscribed from the RTI through the new "Country" class "population" complex data type. Finally, the former Hello World interactions were removed from the federates. This comprised the SAM Federation prototype.

SAM FEDERATION DEVELOPMENT PROCESS

With a clearer understanding of the HLA and the operation of the RTI, we evaluated the requirements for implementing the SAM Fed. Several key requirements included:

- The simulation scenario must execute in a synchronous and sequential manner
- No attribute updates can be missed
- The legacy SAM code cannot be modified
- The federation should operate in real time
- The federation operation must run accurately as verified with legacy simulation results
- The federation must operate across different computer platforms (Sun and WindowsNT).

We then utilized the FEDEP for project development. First, we checked resource repositories for FOMs and SOMs. Many FOMs and SOMs were reviewed but they were judged to be too complex for our needs, or simply did not apply to the scenario. The FOMs were useful in presenting the different design philosophies for organizing class structures, allocating attributes to the classes, and the use of complex data types. Our philosophy presumes that streamlined federates will yield themselves to easier incorporation into other federations.

The SAM Fed FOM and SOMs were defined through an approach that assigned persons with detailed knowledge about the SAM systems to represent the needs of each federate. We utilized the Object Model Development Tool (OMDT) to build the SOMs for each of the federates on an attribute by attribute basis. Lessons learned from our SAM prototype development effort proved invaluable for decisions involving class and attribute specifications. Separate classes were defined for each SAM function and specific attributes were grouped into complex data types such as x, y, z position data into a position complex data type. Each attribute could only be used if it was deemed necessary for the federate or for the federation. Building from the prototype implementation, SIMS was leveraged to provide a means of incorporating all FMIC SAM models as federation participants. We scrutinized the critical SIMS parameters and recorded them in the appropriate SOMS via the OMDT. When the SOMs were defined for the target, radar, SAM and controller federates, the FOM was defined. We specified the FOM by the intersections of the SOM attributes.

With a working prototype federation at our disposal, the key requirements were verified as attainable. All attributes need to be regulated and constrained to ensure updates are not missed. However, the real time operation through the single federation executive process within the RTI 1.0.3 proved challenging. Experimentation with the SAM prototype federation indicated a tighter constraint on the update rate than the data volume, or throughput. We devised a short term solution of bundling to improve the federations real

time performance. The data bundling concept is comprised of five time updates packaged together as a single attribute. Each of the federates could then process each of the five time steps, then populate each of the corresponding five updates in the bundled output attribute to be processed by the next federate. The bundling of updates is made possible since the federation is an open loop scenario.

SAM FEDERATION IMPLEMENTATION

In summary, we began the SAM Fed implementation by modifying and adding to the federate ambassador portions of code in each of the prototype federates to define the complex attribute types specified in the FOM. We enumerated the attributes and implemented handle requests for the attributes. Next, we expanded the prototype federate ambassador code to specify the attribute receive and attribute publish functions for all of the new attributes. To specify the new attributes, we modified the publish and subscribe functions. We modified the federate simulation loops to extract the SIMS variables from the attributes and to map and bundle the federates newly computed variables to the attributes. We then modified each of the federates to process each of the five time steps, then populate each of the corresponding five updates in the bundled output attribute to be processed by the next federate. We then generated a byte-swapping routine to correct for the endian differences between the Pentium class PCs and the SUN systems.

It is important to note that the source code for all of the federates was combined into a single source file. We added logic to the code to specify each federate by environment variables. This reduced the burden of configuration control of the source code across multiple machines within the DTC lab. In addition, any federate could be run on any platform by simply specifying the appropriate environment variables at compile time and runtime. An environment variable was also added to specify the platform type, Sun or PC, in order to utilize byte-swapping routines and to correct function call differences. As a follow on effort, we created two remote observer/plot federates to subscribe to data of interest and generate graphical engineering level displays. The remote observer/plot federates run under the WindowsNT and Windows95 operating systems.

TESTING

We performed testing of the SAM Fed by direct comparisons to the legacy PCDEMO executive and generic SAM model results. The test results from the SAM Fed directly match the PCDEMO test results.

Preliminary testing of the FMIC SAM models within the SAM Fed have also been performed. The FMIC SAM models were each tested by recompiling the SAM model federate with the new SAM missile source code. The SAM model federate then called the FMIC SAM missile code in the same manner as the unclassified generic SAM missile

code. In addition to verifying the operation of several FMIC SAM models within the SAM Fed, the preliminary testing has provided validation of our plug-and-play federate concept.

We performed tests across different platforms to verify the SAM Fed byte swapping routines functioned correctly. We also tested the operation of the SAM Fed across wide networks on a limited basis.

PERFORMANCE

No diagnostic performance measures have been performed a this time. However, several subjective observations have been made with respect the SAM Fed and the RTI versions 1.0-3 thru 1.3-4. The SAM Fed will run in real time when hosted on the Sun platform. When the SAM Fed is distributed between WindowsNT systems with the RTI and Federation executive hosted on a WindowsNT PC, the SAM Fed runs slower than real time (approximately 7 seconds of simulation time per 10 seconds of wall clock time).

A combination of factors contributed to the performance of the SAM Fed. The processor speed was a major contributor to the federation execution speed. However, we experienced significant performance impact with variations in the available RAM and network card speeds.

The platform configurations hosting the SAM Fed include:

- Dual 300MHz PentiumII, 256Mb RAM, 100Mbps network card
- 300MHz PentiumII, 64Mb RAM, 100Mbps network card
- 166MHz PentiumII, 196Mb RAM, 10-100Mbps network card
- 296MHz Sun Ultra2, 124Mb RAM, 100Mbps network card.

Experimentation with the tick function indicated the function was not working as specified in the *High Level Architecture Run-Time Infrastructure Programmers Guide*[2]. The maximum tick period seemed to be ignored by the RTI 1.0-3 where it took the time it needed to perform its own processing. Note that tests with SAM Fed and RTI version 1.3-4 indicate the tick function works correctly.

HLA CERTIFICATION

The SAM Model Federate, version 1.0, was certified HLA Compliant by DMSO on 11 May 1998. The SAM Model Federate was the fourth federate ever certified, and the first Air Force federate certified. The migration of the legacy model set is not yet complete. However, the SAM model federate is in the process of being tested with all of the FMIC missile models. When the testing is completed all FMIC missile models will be declared HLA compliant.

LESSONS LEARNED

As with any new development effort, there were many lessons learned from the SAM Fed project. Some of the most significant are listed below.

- We conducted the SAM Fed design and development with more confidence due to our experiences with the prototyping efforts. The prototype development approach allowed for experimentation on a functional federation early in the project.
- Lots of memory, RAM, is necessary for fast execution times.
- Fast ethernet, 100Mbit, is necessary where high update rates are utilized (greater than 60 Hz).
- Minimalist design approach from the beginning of the project helped to ensure meaningful communication was managed correctly, and minimized the overhead of seldomly used variables.
- The use of a single homogeneous source code that delineates between the federates by setting environment variables at runtime simplified configuration control. In addition, the federates are easy to execute via short batch files or shell scripts. The end product is capable of running one or all federates from the same code.
- Building a federation from legacy code proved very beneficial in both the implementation phase as well as the testing phase of development. The federation results are easily verifiable when compared to the legacy results.
- Configuration control is a must with any development project. Clear and easy to follow control procedures for multiple member development teams help ensure the procedures are followed.
- In the design of a real time federation, attempts should be made to keep high data rates off the RTI and contained within individual federates.

LIVE RANGE PARTICIPANTS

We are currently studying the feasibility of implementing a live test range federate. We have capitalized on our SAM Fed efforts by replacing the target federate with a live test range federate to provide time, space, and position (TSPI) information as input to the federation in real time. Feasibility tests with the SAM Fed have been performed with live aircraft on the Eglin test range via a real time TSPI data processor.

REMOTE OBSERVER FEDERATE

Remote viewer federates provide a means of receiving federation data in an unobtrusive manner for use in plotters, loggers, and visualizers. We have developed two remote viewers for use with the SAM Fed. Our Plot federate produces a plot of the target and missile objects over a North versus East plot and an Altitude versus Downrange plot. The Plot federate provides clear situational awareness when observing the SAM Fed

execution. We developed the Engineering Unit Display (EUD) federate to provide a real time listing of all SAM Fed FOM attributes to the screen. The EUD has proven to be a valued diagnostic and evaluation tool. Since each of the remote viewer federates run as a single user-interface thread, they can run under WindowsNT or Windows95.

FUTURE ENHANCEMENTS

Some of our planned future enhancements include:

1. We plan to integrate test range assets as SAM federation participants.
2. We plan to continue investigation of ways to utilize new versions of the RTI to enable us to abandon our federate data 'bundling' concept, while still maintaining real time performance.

REFERENCES

[1] Air Force Development Test Center, 96th Communications Group, Systems Technology Division, Modeling and Simulation Support Flight, Development and Analysis Section: Standard Interface for Missile Simulations (SIMS), 29 July 1995.

[2] DMSO, Virtual Technology Corporation, SAIC: Programming with the Run Time Infrastructure 1.0, Hands-On Practicum course notes, 6 October 1997.

[3] DMSO, MITRE, SAIC, Virtual Technology Corporation: High Level Architecture Run-Time Infrastructure Programmer's Guide, Version 1.0, 15 May 1997.

AUTHOR BIOGRAPHY

Jason Lucas is a Senior Aerospace Engineer for TYBRIN Corporation in support of the 96th Communications Group, Modeling and Simulation Support Flight, Development and Analysis Section (96CG/SCTOS) at Eglin AFB, FL. Jason develops and manages surface-to-air missile simulations and antiaircraft artillery simulations for use in electronic countermeasures and munitions survivability testing. His HLA background includes the 1997 Fall Simulation Interoperability Workshop (SIW), the Hands-On Practicum, the 1998 Spring SIW, and the 1998 Fall SIW.

Bob Kent is a Computer Scientist in the 96th Communications Group, Modeling and Simulation Support Flight, Development and Analysis Section (96CG/SCTOS) at Eglin AFB, FL. Bob has 21+ years of experience (19 in government service) and has programmed extensively using the following languages, protocols, and environments:

(1) FORTRAN and DECWindows on Digital VAX/VMS
(2) FORTRAN, C, Xwindows on SUN Solaris
(3) FORTRAN, C, Xwindows, and GL on SGI IRIX
(4) FORTRAN, C, QuickWin, and MFC on Microsoft Windows95/NT.

His HLA background includes the 1997 Fall SIW, the Hands-On Practicum, the 1998 Spring SIW, the HLA Focused Training – HLA Compliance testing, the 1998 Fall SIW, and the 1999 Spring SIW.