Design and Implementation of a Lunar Communications Satellite and Server for the 2012 SISO Smackdown

Dennis Bulgatz
Analytical Mechanics Associates
1500 Perimeter Parkway
Huntsville, AL 35806
(256) 830-4811, ext. 111
bulgatz@ama-inc.com

Daniel Heater
Daniel A. O'Neil
Marshall Space Flight Center
Huntsville, AL 35811
(256) 544-1302, (256) 544-5405
daniel.heater@nasa.gov
daniel.a.oneil@nasa.gov

Bryan Norris
Science Applications International Corporation
Building 5400, Room E101
Redstone Arsenal, AL 35898
Bryan.C.Norris@saic.com

Bradley C. Schricker
Dynetics, Inc.
1002 Explorer Blvd.
Huntsville, AL 35806
(256) 964-4979
brad.schricker@dynetics.com

Abstract: *Last year, the Simulation Interoperability Standards Organization (SISO) inaugurated the now annual High Level Architecture (HLA) Smackdown at the Spring Simulation Interoperability Workshop (SIW). A primary objective of the Smackdown event is to provide college students with hands-on experience in the High Level Architecture (HLA). The University of Alabama in Huntsville (UAHuntsville) fielded teams in 2011 and 2012. Both the 2011 and 2012 smackdown scenarios were a lunar resupply mission. The 2012 UAHuntsville fielded four federates: a communications network Federate called Lunar Communications and Navigation Satellite Service (LCANServ) for sending and receiving messages, a Lunar Satellite Constellation (LCANSat) to put in place radios needed by the communications network for Line-Of-Sight communication calculations, and 3D graphical displays of the orbiting satellites and a 3D visualization of the lunar surface activities. This paper concentrates on the first two federates by describing the functions, algorithms, the modular FOM, experiences, lessons learned and recommendations for future Smackdown events.*

1. Introduction

Responding to the needs of industry and the National Aeronautics and Space Administration (NASA), the Simulation Interoperability Standards Organization (SISO) initiated an annual distributed simulation event. The ***SISO Simulation Smackdown*** is an international, cooperative experience where teams of university students – with help from faculty advisors, modeling and simulation (M&S) professionals within industry, NASA, and other areas of government – build and participate in a simulated lunar resupply mission [1]. This event provides college students with hands-on experience in the development of distributed simulations using the High Level Architecture (HLA).

Participating universities in the 2012 Smackdown simulation included the University of Alabama in Huntsville (UAHuntsville), Massachusetts Institute of Technology (MIT), Penn State University, universities from Genoa, Pisa, and Rome, Italy, and Technion University in Israel.

Technion contributed an Object Process Modeling (OPM) application [2]. Penn State developed simulations of a cargo landing vehicle and a cargo transfer rover. The MIT team produced simulations of a mobile resource utilization plant and a scouting hopper that jumped from one place to another in search of minerals for the mobile resource utilization plant. The three universities from Italy formed two teams, respectively responsible for: (1) a simulation of an asteroid tracking system, and (2) a simulation of a warehouse and inventory management system for lunar surface exploration system parts and consumables..

NASA's Johnson Space Center (JSC) contributed an environment simulation that provided reference frames of the Sun, Earth, and Moon and a one-second ***heartbeat*** for the distributed simulation. The orbiting space-craft was another federate developed by JSC. The Central Run-time Component (CRC) that managed the distributed simulation, the Virtual Private Network (VPN) for remote participation, and the local area network for integrating participating computers at the conference were operated by the JSC team.

The UAHuntsville team contributed four federates: (1) a radio communications server, (2) a satellite federate with an orbital propagator and radio, (3) a 3D graphics display of a constellation of four communications satellites, and (4) a 3D visualization of the lunar resupply mission. This paper explains the design of the first two federates.

The UAHuntsville team decided upon a four-satellite constellation with an elliptical orbit that ensured a significant amount of ***hang-time*** over Hadley Rille, but not continuous coverage so that line-of-site (LOS) and message relay interactions would be relevant.

The communications model consisted of two basic submodels – ***radios*** and ***routers***. Radios simply received and transmitted messages, while routers had those capabilities plus the ability to serve as a hub between two communications points that do not share unobstructed line-of-sight. Initially, a highly-detailed message format was considered in an attempt to closely replicate realistic message traffic between vehicles in a lunar resupply mission. However, to support the more immediate need to serve as a proof-of-concept, the team decided to simplify the message format, expecting that a simpler model would encourage greater use of the radio software by other Smackdown participants. Thus, the message format consisted of basic header for routing within the HLA federation and a payload. The following section provides a low-level description of the communications satellite federate.

2. Communications Server Federate Implementation

With the understanding that the SISO Smackdown depends on the collaboration of student teams from multiple universities, a primary goal of the radio communications system design was to minimize the engineering impact on other teams in an effort to increase the likelihood that other teams would use the radio communications service. After initial considerations to implement radios as libraries that could be provided to other teams for integration – and realizing that this would generate the need for Java, C++, and MATLAB versions – the UAHuntsville team determined that implementing much of the radio system as a service would reduce the implementation burden on other teams. Still, the Java source code for an example radio was made available to all teams.

Messages in the radio system were defined as HLA interactions with ***transmit*** (TX) and ***receive*** (RX) interactions being defined separately. A TX interaction would be a message from a radio to the communication service and an RX interaction would pass a message from the communication service to the receiving radio. This division allows for the communication service to subscribe to only TX interactions and publish only RX interactions while the radios subscribe only to RX interactions and publish only TX interactions. This division made the logic for sending and receiving messages clean, as there was no need to decode the interaction to determine direction.

Storing the message destination in the HLA header allowed radios to optionally use HLA Data Distribution Management (DDM) capabilities to filter message traffic on the destination of an RX interaction so that a message recipient would only receive messages that were destined for it. This combination of direction-oriented interactions and DDM meant that bandwidth use could be minimized for the Run-Time Infrastructures (RTIs) that supported DDM on the sender-side of the interaction,

a critical consideration in any distributed simulation environment. In defining the radio Federation Object Model (FOM), it was initially intended to encode the position of the transmission source in the interaction. Paul Grogan, a student at MIT and contributor to the Smackdown, suggested a further improvement to conserve bandwidth. As a message store/forward capability within the communication service was intended, it was determined that the communications service would need to know the position/location of entities to determine line-of-sight (LOS) – information that was already being published. In this context, LOS is considered a Boolean value indicating whether a straight, unobstructed line existed between two points in space (*true*) or not (*false*). This concept is illustrated in **Figure 1.** Line of sight algorithm[3]. **Figure 2** depicts the architecture of the communications server.

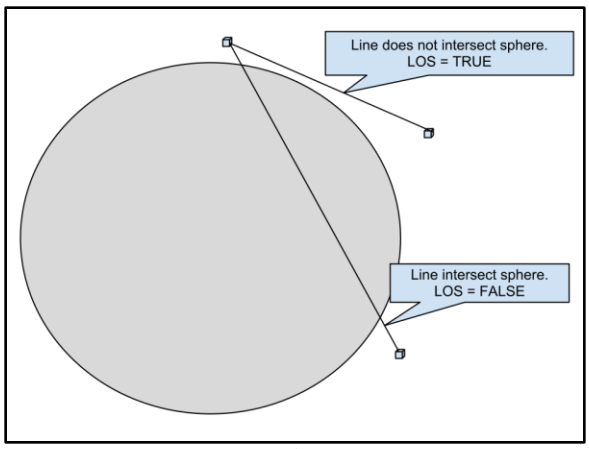

Figure 1. Line of sight algorithm

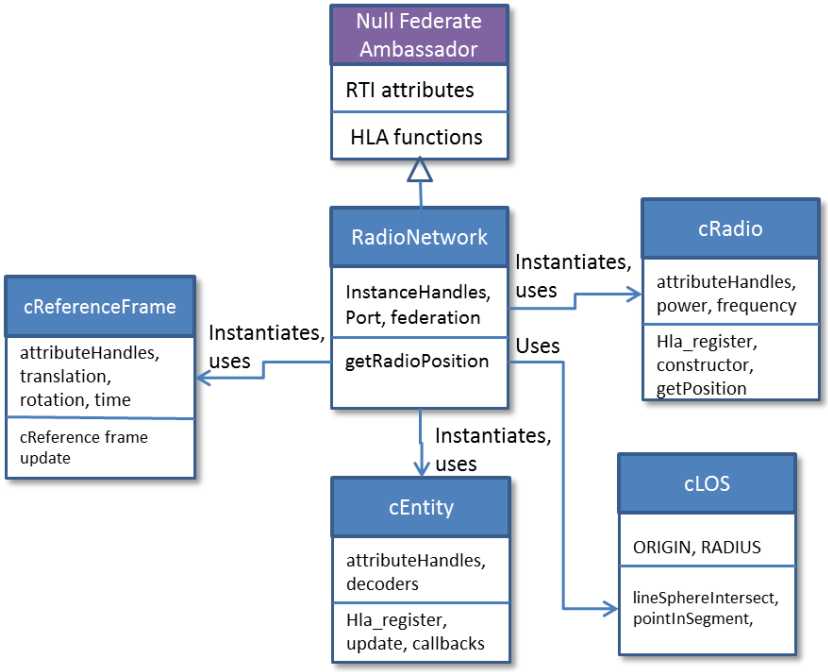

Figure 2. Simplified Class Diagram of LCANServ

The position of transmission from the radio object attributes was omitted, and instead only an elevation relative to the base position of the entity owning the radio was included in the transmission. This elevation was deemed necessary, as entities on the lunar surface would be operating as though the terrain was a flat plane; however, the communication service modeled the lunar body as a sphere for the purpose of the LOS calculations. Typically, surface entities would report their position as being on the lunar surface or even below the mean lunar radius, based upon the diameter of the sphere used for LOS calculations, as the scenario took place at Hadley Rille, 1800 meters below the mean lunar radius. Such a position would never be considered within LOS to other entities since the LOS algorithm would consider the radio to be ***inside*** the moon, as opposed to somewhere on the surface. Instead of using the reported position, the radios' reported elevations were added to the mean lunar radius so that radios were correctly considered to be above the lunar surface. This approach allowed the radios to have reasonable range to the horizon for LOS.

The LOS algorithm needed to have a mechanism for determining whether a line between the source and the destination radios intersected the moon's surface. If a line between the two points intersected the spherical model of the moon, it was required to determine if the intersection occurred between the two points. Consider this surface to surface LOS situation illustrated in **Figure 3.**

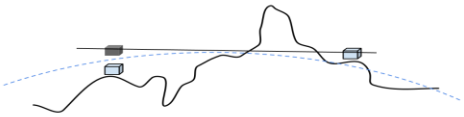

Figure 3. Surface to surface LOS

First, the entity on the left is below the mean Lunar radius (blue dashed arc) so the elevation parameter supplied in the radio attributes is used to change its effective position to a value above the radius of the sphere. This new effective position is represented by the grey box. Next, the straight line connecting the entities is considered. It does not intersect the sphere, therefore LOS is true. Note though, that if the intersection had been between the two entities, then LOS would be false. This algorithm does not determine the actual intersection point(s).

Notice in **Figure 3.** Surface to surface LOS, that there is a hill blocking LOS. This level of fidelity was not accounted for in LCANServ, as no terrain database was used for the Smackdown. The entities operated as though they were on a flat plane, and LCANServ projected that flat plane onto a perfect sphere.

The radio FOM used for the Smackdown event (Smack_radio FOM) defines for format for HLA federation messages. **Figure 4.** Radio modular FOM depicts the modular FOM for the radio.

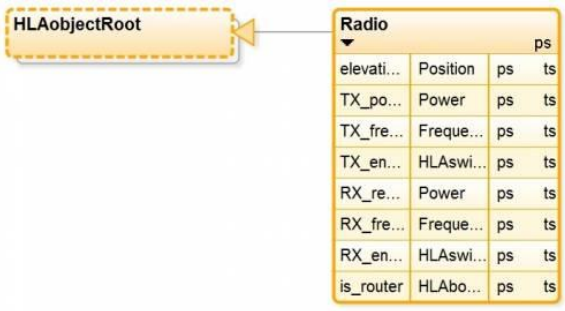

Figure 4. Radio modular FOM

Note that this message does not specify any protocol such as the Consultative Committee for Space Data Systems (CCSDS) or TCP/IP that would typically be used by a space communications network. Instead, it is left to the users of the system to negotiate message headers, timestamps, Cyclic Redundancy Checks (CRCs), and the like.

The communication network does support ***routing radios***, which can store a received message and forward that message to the specified destination at a later time. Messages can be marked as ***broadcast***, which indicates that the message will be received only by radios currently in range and should not be stored for later forwarding. **Figure 5.** Radio message presents the message object model template. As opposed to the general implementation in the real world, message store/forwarding is ***not*** implemented by passing messages between radios. The list of messages in route is stored and handled by the communication server. Thus, a satellite may effectively be used to route a message, but the message is never sent to that satellite federate. This reduces the necessary distributed simulation bandwidth and allows the store/forwarding logic to be implemented without the prohibitive number of interactions and housekeeping mechanisms that would otherwise be required for such a system. To make a determination as to whether a communication interaction was successful, the following information needed to be known:

- Transmitter position/location
- Transmitter power
- Transmission frequency
- Required receive signal strength
- Receiver frequency

Since atmospheric effects were unaccounted for, a determination of whether two radios were in range of one another was performed using the free space path loss equation [4].

A proposed enhancement of the communication service is to add navigation services. Research was conducted regarding proposed systems for lunar navigation. The concept that seemed most promising was a technique called ***liaison navigation***. The concept is explored in a Ph.D. thesis by Keric Hill of the University of Colorado entitled, "Autonomous Navigation in Liberation Point Orbits," published in 2007. A number of other published papers by the same author and his advisers are also available [5].

Figure 5. Radio message object model template

Liaison navigation takes advantage of the uniqueness of the gravitational forces exerted on a satellite in a halo orbit from the three-body, Earth-Moon-Satellite system. The authors' research predicts that the absolute position and velocity of two spacecraft can simultaneously be estimated autonomously with no Earth-based tracking using crosslink range measurements between the spacecraft. In simulations accounting for effects of planetary ephemeris, solar and lunar gravity, and solar radiation pressure, the author indicates that, "position errors for lunar spacecraft were on the order of 10m RSS and about 100m RSS for halo orbiters in various locations."

3. Lunar Communications Satellite (LCANSat) Federate Implementation

The LCANSat Federate simulates a constellation of one or more satellites orbiting the Moon. LCANSat is a Java application, developed with Java version 6 (1.6 JRE). The architecture is object-oriented and designed to organize the code into re-usable and easily understandable blocks.

The ***Driver*** class contains the main method. From the terminal, the user is able to enter the IP address of the Federation CRC. The Driver object calls each Satellite object's orbital propagator method to update the position. With each iteration, the Driver polls the UserIO object to determine whether the user has entered a dot (".") at the command line interface. The main method in the Driver class orchestrates the following:

- Accepts input from the terminal for the IP address of the CRC
- Creates a federate object to persist the HLA federate details
    - The federate creates a Connection Object which persists information about the federation's connection to the federation
    - The federate handles time advance grants and the HLA call backs
- Instantiates a constellation of satellites and a radio for each satellite instance using constructors in the radio and satellite classes
- Creates a User IO thread to display all radio messages and allow creation of radio messages from the terminal
- Sets a Boolean for time management variable, ***isHLAContrained***
- Moves the satellites in orbit by incrementing the true anomaly a fixed amount each time step

Figure 6. The LCANSat Software Architecture. The UserIO class implements an executable thread, not tied to simulation time. Since radio messages are interactions, which are not time dependent, they can be sent and received at any time. The UserIO class initializes the HLA attribute set for a radio message and sends a request to the RTIAmbassador to both publish and subscribe to radio messages. Finally, the UserIO class contains the necessary encoders and decoders for sending and receiving messages.

The Federation class contains a constructor to build a federate. Attributes passed to the constructor include the RTI host IP address, simulation time interval (one second) and the HLA_TIME_CONSTRAINED Boolean. The federate creates a connection to the CRC as part of the constructor.

In addition, the federate stores the RTI ambassador, encoder factory, and other HLA related objects. The Federate also contains all the HLA callbacks used, such as RequestTimeAdvance and registerObjectInstance.

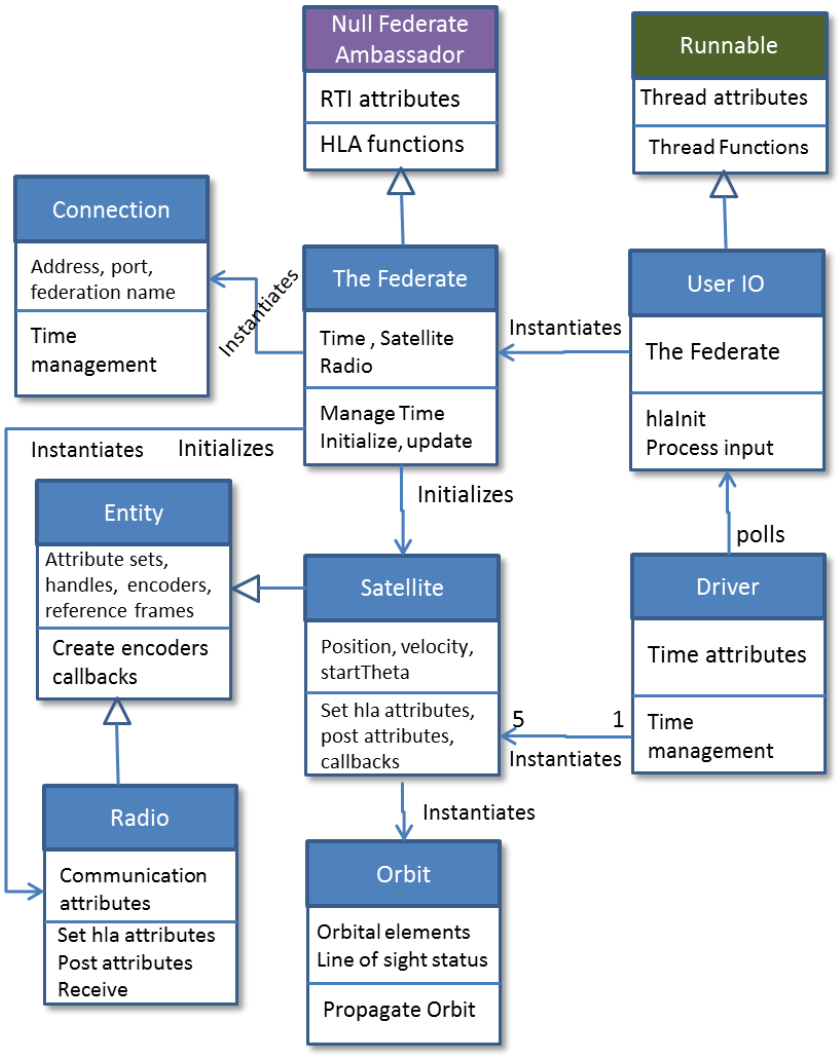

Figure 6. The LCANSat Software Architecture

The Connection object, instantiated by TheFederation object, specifies the IP address and port of the CRC and creates or joins the federation. TheConnection also defines the FOM modules used by the Federate. FOMS used include:

- SISO_Smackdown_1011_core.xml
- SISO_Smackdown_1011_environ.xml
- SISO_Smackdown_1011_entity.xml
- Smack_radio.xml

Satellites and radios were both modeled in the HLA FOM as a ***PhysicalEntity***. To reuse code, an Entity class was created to manage all the common properties and attributes of a radio and satellite.

Satellite objects are instantiated by the Driver class, using an instance of theFederate (instantiated in the Driver) and a unique name. The satellite constructor creates an default Orbit for each Satellite, with the starting angle (true anomaly) mapped from the name of the satellite (Satellite.getStartTheta), staggering the four primary satellites 90 degrees apart.

The default Orbit object, which determines the path of the satellite, was adjusted by experimentation with the obit settings resulted in an elliptical orbit that provides a lot of hang time over Hadley Rille to maximize the time each satellite is in-view of the surface assets. The orbital propagator uses a similar line-of-sight algorithm as the communications server. When a satellite is in view, the Orbiter object reports the status and the satellite object adds the status to the attribute set for update.

Figure 7 Orbital Trajectoryillustrates the orbital path (magenta) and a plane to define the LOS boundary between the lunar base (blue dot) and orbital path. This is a similar approach to what is

used in LCANServ. The LOS boundary plane (yellow) is the tangential plane through which the lunar base location passes and whose normal is collinear to a line from the center of moon to the base (radius vector).

Orbits were defined using semi-major axis, ascending node, argument of perigee, orbital inclination, eccentricity, initial time of perigee passage, and true anomaly. Propagation of the orbits was done by incrementing the true anomaly for each orbit by a fixed amount each time step [6].

Given the Cartesian coordinates of the lunar base are $BASE_X$, $BASE_Y$, and $BASE_Z$, and the radius of the moon is $radius_m$, the distance from the plane to an point on the orbital path (x,y,z) is defined as:

$$distance = (x * BASE_X + y * BASE_Y + z * BASE_Z - radius_m^2) / radius_m$$

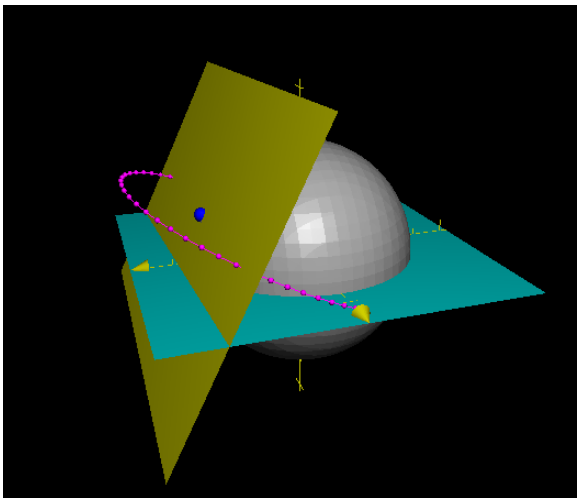

Figure 7 Orbital Trajectory

When *distance* is greater than zero, the point on the orbital path is in LOS of the moon base. LCANSat broadcasts the LOS status via the ***status*** attribute on the PhysicalEntity Class.

4. Collaborative Creation of Virtual Environments

January 2012 saw the first offering of the "Collaborative Creation of Virtual Environments" course at UAHuntsville. Mikel D. Petty, Ph.D., invited NASA personnel and RTI vendors to provide to the class a variety of perspectives of distributed simulation and the HLA standard [8]. Topics of the invited presentations included:

- Satellite systems design
- Orbital trajectory analysis
- Simulation scenario development
- A NASA application of HLA
- Virtual Private Networks (VPN)
- ForwardSim HLA Tool Kit
- SISO Smackdown objectives
- VT MAK RTI
- Pitch RTI

Presentations were given via WebEx and teleconference. Other university teams were invited to participate in the discussions. The guest lectures provided an introduction in spacecraft system design and distributed simulation infrastructure.

5. Development Process

A few development decisions faced by the UAHuntsville team included programming language selection, development tool selection, RTI versions, and implementation of time management within the federates. After reviewing the available code examples and team members' experience, the team chose Java for developing the LCANSat and LCANServ federates.

The EZ Button Federate, developed by Dr. Edwin "Zack" Crues, served as a template for the time management callbacks and an example for reference frames. The NASA team at JSC deployed an Environment federate, which published positions and orientations of reference frames for the sun, earth, and moon. Additionally, the Environment federate published the heartbeat for the SISO Smackdown for coordinating the federation.

The team used the Netbeans and Eclipse Integrated Development Environments (IDE) for code development and a Google Code repository for configuration management. A Google site was used to manage files and links.

Both VT MÄK and Pitch offer free community editions of their RTIs, which enabled development of two federates. Due to new releases by both companies, the free versions were incompatible with the CRCs on the Smackdown simulation server. This incompatibility required all of the federate developers to obtain the licensed version of the Local RTI Component (LRC).

6. Lessons Learned

Perhaps, the most crucial lesson learned pertains to the end goal – that is, future teams should begin with the end in mind by planning the simulation scenario. The actual SISO Smackdown simulation event lasted for roughly ninety minutes. During the event, an emcee introduced the team leaders who described the federates developed by their teams. Following the team introductions was an explanation of the mission scenario. The last part of the event was a presentation of awards to the teams. Meanwhile, the distributed simulation federation ran on a local area network and a VPN. Significant simulation events such as the touch down or lift-off the cargo landing vehicle were announced so the audience could focus their attention on the event. Ultimately, each team ought to plan a scenario so that an important simulation event occurs within those ninety minutes, preferably staggered with other teams' simulation events.

An example of a significant event for the communications system could be the transmission of a message from a scouting hopper to a mobile resource plant. Another example could be the transmission of a near-miss warning from an asteroid tracking system to the orbiting space-craft and cargo lander, which would involve federates from all of the SISO Smackdown simulation participants.

Designing a system and learning the HLA standard simultaneously can be difficult, especially if team members are enrolled in multiple classes or have full-time jobs. If a SISO Smackdown Simulation team can work with an engineering design team it will allow the simulation team to focus on implementation of the federates. Learning the HLA standard prior to the start of the calendar year may enable teams to be productive during the month of January. Suggestions for learning the standard include stepping through example modular FOMs and federate code while reading the IEEE 1516 standard. Reading the standard in one window while stepping through an XML or source-code file in another window can help to illustrate the explanations in the standard.

Suggestions for a development process include requesting RTI licenses from both vendors for every team member, conducting a team walk-through of example code, identifying a systems integrator, and selecting development and configuration management tools early. Obtaining licenses for all team members will ensure consistency in computer configurations. Writing detailed procedures, capturing screen shots for configuring files, and posting the documentation on a team web-site or on the Smackdown Wiki can also assist in communication. Conducting walk-throughs of example code can also help participating team members to learn the standard together.

Establishing a communications plan between team members early in the process can facilitate intra-team communication. If team members are willing to post their phone numbers on an internal team website, building a contact list so people can communicate by phone can also help. Skype is an excellent application for programming in pairs. The free service from Skype provides screen-sharing, so one team member can watch while another team member writes code. However, it was discovered that the Sonic Wall Net Extender prohibits access to the rest of the Internet, so to test a federate using Skype, one of the Skype participants needs to run a CRC.

Given the considerable challenges encountered during integration and test activities, it may be valuable to consider meeting two times per week during the integration and test phases. Teams can be highly productive with two work sessions per week at two or three hours per session. Coordinating times for a large team can be difficult so consider working in pairs or groups of three.

Development activities can be improved by requesting the licenses for all of the team members. If an LRC is used, support tools are unavailable. A detail that required attention each time someone logged into the VPN was the assigned Internet Protocol (IP) address assigned by the Sonic Wall Net Extender. Without the support tools, the developers had to identify the assigned IP address in a configuration file after logging into the VPN. A misconfiguration of a VPN data file caused subsequent federates, attempting to join the federation, to fail. With the support tool provided with the licensed system, the developer can select an IP address from a drop-down list on a graphical user interface.

Multiple federates can share a common code base. When designing multiple federates, it is sensible to look for common classes and manage the configuration of the code-base. Spending time upfront on a good object-oriented design can save development time and the effort associated with configuration management of multiple code-bases.

7. Conclusions and Recommendations

The SISO Smackdown Simulation event provides an opportunity to list experience in development of an international distributed simulation on a résumé. The previous section described ways to get the most out of the experience. Recommendations for potential products that could improve future Smackdown development activities include an open-source Environment federate, a reusable code library with well documented examples, a developer's handbook, and a catalog of concepts.

The current models that provide reference time and reference frame updates are implemented by JSC and are not currently available for public release. This creates a dependency on the JSC servers and VPN to conduct integration testing. The free open source Java Astrodynamics Toolkit could provide the source code for an open Environment federate [9]. An independent study project could produce a federate that generates a one hertz heart-beat and publishes reference frames for the sun, earth, and moon. Participating teams could use the open-source environment federate to establish their own servers for development and testing.

A reusable code library with well-documented examples, especially time-constraint examples, can help new teams climb the learning curve quickly. Posting archives of Java, C++, and MATLAB examples on the SISO Smackdown website can be part of a starter kit. Java Doc can generate web-based documentation from specially annotated comments in the code.

Another desirable item in a starter kit is a developer's hand-book. While the IEEE-1516 standard is useful as a reference after someone understands the standard, it is not a fast read or a novice-friendly document. A developer's handbook could help novices by presenting flow-charts, example code, and detailed explanations.

A catalog of concepts can help new teams get ideas for needed federates. Previously developed federates could be described in a web-based concept catalog with an artist's concept image, a brief paragraph of the federate's capabilities, and a link to a code repository. Other system concepts that have not been implemented could include an image, a description, and notional scenarios that illustrate how the federate could interact with other federates in the simulation.

References

[1] Elfrey, Priscilla, "About", SISO Smackdown, 2011, http://sisosmackdown.com/about/, viewed on May 5, 2012.

[2] Dori, Dov, *Enterprise Systems Modeling Laboratory*, NASA Smackdown Project. Technion U, n.d., http://esml.iem.technion.ac.il/?page_id=481, viewed on May 5, 2012.

[3] Bourke, Paul, "Intersection of a Line Anda Sphere (or Circle)," Nov. 1992, http://paulbourke.net/geometry/sphereline/, viewed on May 5, 2012.

[4] Kou, Yajun. "Derivation the dB version of the Path Loss Equation for Free Space." Electrical & Computer Engineering, University of Victoria, September 19, 2000. http://www.ece.uvic.ca/~peterd/35001/ass1a/node1.html, viewed on May 24, 2012.

[5] Hill, Keric, "Autonomous Navigation in Libration Point Orbits." *Diss. University of Colorado*, 2007,http://ccar.colorado.edu/geryon/papers/Misc/Hill_thesis.pdf, viewed on April 19, 2012.

[6] "Johannes Kepler: The Laws of Planetary Motion", Astronomy 161 The Solar System, Dept. Physics & Astronomy University of Tennessee, August 10, 2000. http://csep10.phys.utk.edu/astr161/lect/history/kepler.html, viewed on May 24, 2012.

[7] Logsdon, Tom. *Orbital Mechanics: Theory and Applications*, Wiley-Interscience, October 24, 1997.

[8] Petty, Mikel D., Collaborative Creation of Virtual Environments, University of Alabama in Huntsville, Spring 2012, http://cmsa.uah.edu/petty/mod796, viewed on May 5, 2012.

[9] Berthold,Tobias. "Welcome to JAT", jat.sourceforge.net, n.d., http://jat.sourceforge.net/, viewed on May 5, 2012.

Author Biographies

DENNIS BULGATZ is a Software Architect and IT Manager at Analytical Mechanics Associates. Since 2002, Mr. Bulgatz has been supported various NASA IT projects, including collaborative environments, web-based applications and data integration at across the NASA enterprise. Prior to 2002, Mr. Bulgatz worked for Automotive and Industrial Automation firms doing design and analysis of electromagnetic variable valve timing systems, loud speakers, voice coil motors, solenoids, fuel injectors, and extendable conveyors. Mr. Bulgatz holds a BSME from UC Berkeley (1988) and an MBA from William and Mary (2000), and is currently taking classes for fun at University of Alabama in Huntsville.

DANIEL L. HEATER is a software engineer developing simulations and flight software for NASA Marshall Space Flight Center. Mr. Heater received a Bachelor of Science Degree from Athens State University in 1998 and is currently pursuing a Masters degree in Modeling and Simulation at the University of Alabama in Huntsville.

BRYAN NORRIS is a Simulation Software Engineer with SAIC, currently technical lead for virtual avionics and Crew Station Working Group support for the Cargo Helicopters Project Office of the United States Army. He has eight years of experience in Software Engineering and Simulation, focusing on visual prototyping, presentation, and simulation. Mr. Norris received his Bachelor of Science degree in Computer Science from the University of Alabama in Huntsville in August of 2003. Mr. Norris received a Master of Science in Software Engineering and a Post-bachelor Certificate in Modeling and Simulation from the University of Alabama in Huntsville in May of 2009. He is currently pursuing a Ph.D. in Modeling and Simulation at the University of Alabama in Huntsville.

DANIEL O'NEIL is a Technical Manager working in the Center Strategic Development and Integration office within the Office of Strategic Analysis and Communication at NASA Marshall Space Flight Center. Mr. O'Neil has over 25 years of Modeling and Simulation experience, supporting Military and Space programs; his contributions include: a real-time vertical situation display simulation for a flight trainer, technical direction for development of a robotic assembly team demonstration, integration of an advanced technology lifecycle and analysis system. Mr. O'Neil received a Bachelor of Science degree in Electrical and Computer Engineering from the University of Alabama in Huntsville (UAH) in 1985 and a Master's of Science degree in Engineering Management from UAH in 1998. Currently, he is pursuing a Ph.D. in Modeling and Simulation at the University of Alabama in Huntsville.

BRADLEY SCHRICKER is a Senior Engineer with Dynetics, Inc., currently serving as Project Manager and Technical Lead on multiple Unmanned Aircraft System (UAS) simulation projects in support of the United States Army Aviation and Missile Research, Development and Engineering Center (AMRDEC). He has fourteen years of experience in Modeling and Simulation, focusing his efforts in the areas of distributed simulation, discrete event simulation, virtual environments, and behavior representation. Mr. Schricker received his Bachelor of Science degree in Computer Science with a minor in Mathematics from Florida State University in 1998. Mr. Schricker earned a Master of Science degree in Modeling and Simulation from the University of Central Florida in May of 2007. He is currently working on a Ph.D., in Modeling and Simulation at the University of Alabama in Huntsville and expects to graduate in 2013.