Proceedings of the 2004 Winter Simulation Conference
R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds

IMPLEMENTING THE HIGH LEVEL ARCHITECTURE IN THE VIRTUAL TEST BED

José Sepúlveda, Luis Rabelo and Jaebok Park

Department of Industrial Engineering and Management Systems
University of Central Florida
Orlando, Florida 32816-2993, USA

Frank Riddick

Manufacturing Simulation and Modeling Group
100 Bureau Drive, MS8260
National Institute of Standards and Technology
Gaithersburg, MD 20899-8260

ABSTRACT

The Virtual Test Bed (VTB) is a prototype of a virtual engineering environment to study operations of current and future space vehicles, spaceports, and ranges. The High-Level Architecture (HLA) as defined by the Department of Defense (DoD), is the main environment. The VTB/HLA implementation described here represents different systems that interact in the simulation of a Space Shuttle liftoff. This example implementation displays the collaboration of a simplified version of the Space Shuttle Simulation Model and a simulation of the Launch Scrub Evaluation Model. Spaceports and ranges are complex systems. This VTB framework is a collaborative computing environment that integrates in a seamless fashion simulation models that represent the different stages in the lifecycle of a complex system to improve the complex-system visualization. A complex system is a non-linear system of systems whose interactions bring together interesting emergent properties that are very difficult to visualize and/or study by using the traditional approach of decomposition.

1 BACKGROUND

The VTB has been designed as an architecture to facilitate the integrated execution of different simulation programs with other supporting non-simulation software. The architecture must deal with issues related to the coordination of different hardware platforms, hardware components, and software components. In addition, the architecture must synchronize the timing of the different simulations and coordinate ownership of objects and message exchanges among several simulations that may be running in parallel, each one addressing different mission components.

The VTB project is an evolution of NASA's Intelligent Launch and Range Operations (ILRO) Program at Ames Research Center (ARC) implemented in 2000 to perform initial studies of a test bed with a demonstration (Bardina 2000; Intelligent Systems Project 2000). The objective of the VTB Project is to provide a collaborative computing environment that (1) supports simulation creation, execution, and reuse, and (2) supports the integration of multidisciplinary simulation models representing elements of launch, range, and spaceport operations. The VTB will provide many benefits, such as enabling risk management evaluations of legacy and new vehicle frameworks, providing a technology pipeline for evaluating and implementing new solutions to existing problems, and enabling better knowledge management.

This paper discusses the integration of simulations of spaceport and range operations. This integration will make possible the functional and logical visualization of these two important systems, and will allow engineers to more thoroughly investigate and display simulations of the operational processes required during the lifecycle of a space vehicle.

2 THE VTB/HLA FRAMEWORK

The integration of simulation models is inherently complex, and that complexity expresses itself and must be delt with in different ways. Simulation modeling software is the means for addressing the complexity of the engineering systems being modeled, but the software itself represents a substantial incarnation of complexity. This complexity is due not only to the technical sophistication necessary to create software simulation applications, but also to sub-optimal software-design decisions and limitations imposed due to commercial concerns.

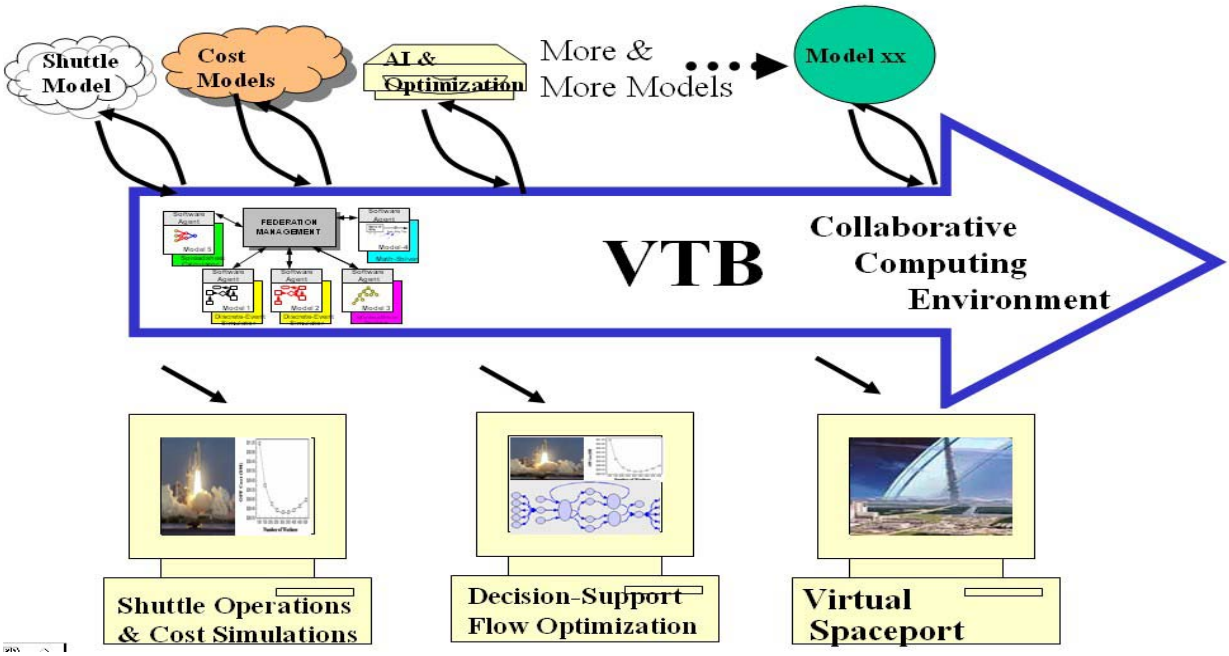

Figure 1 - The Virtual Test Bed Environment

To address the problem of describing this inherently complex integrated simulation system, the VTB/HLA framework will be described in parts. First, a brief description of the HLA will be presented. Next, a description of the VTB and how it can be integrated with the HLA will be presented. This will be followed by an extended example of how the integrated test bed can be used.

2.1 The High-Level Architecture (HLA)

The Department of Defense's intention in creating the HLA was to have a system where existing computer simulations could be combined to address new, more complex, problems of interest.

HLA is formally defined by three components: (1) HLA rules, (2) an interface specification called the Run-time Infrastructure (RTI), and (3) a data specification tool called the object model template (OMT). HLA rules are a set of ten basic principles that define the responsibilities, relationships, and the ways to exchange information among federates and RTI. In the HLA, simulations are called "federates" and a group of federates operating together in a distributed simulation is called a federation. The RTI is the only executable software component of the HLA, and its interface provides services that allow federates to exchange information and coordinate federation execution. The functionality provided by the RTI includes services to manage federation creation and operation; information exchange responsibilities within the federation; object creation, identification, ownership, and deletion; and, time synchronization and coordination. Federates exchange information with other federates by invoking the services of the RTI, and receive information from the RTI through asynchronous callbacks. The OMT defines the structure of information that can be shared by federates in a federation. The key data elements defined by the OMT are objects and interactions. Objects are persistent data entities that are created, modified, and deleted by federates during a federation execution. Interactions are non-persistent data entities that function like messages sent from a federate to one or more other federates. A key function of the OMT is to promote information sharing and simulation reuse (Kuhl et al. 1998).

The VTB is implemented on top of these components. In addition, it has to support and incorporate the capability to integrate applications that support other distributed computing approaches, such as the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA), the World Wide Web Consortium's (W3C) Simple Object Access Protocol (SOAP), and Microsoft's Distributed Common Object Model (DCOM).

2.2 The Virtual Test Bed (VTB)

At its heart the VTB is a cooperative computing

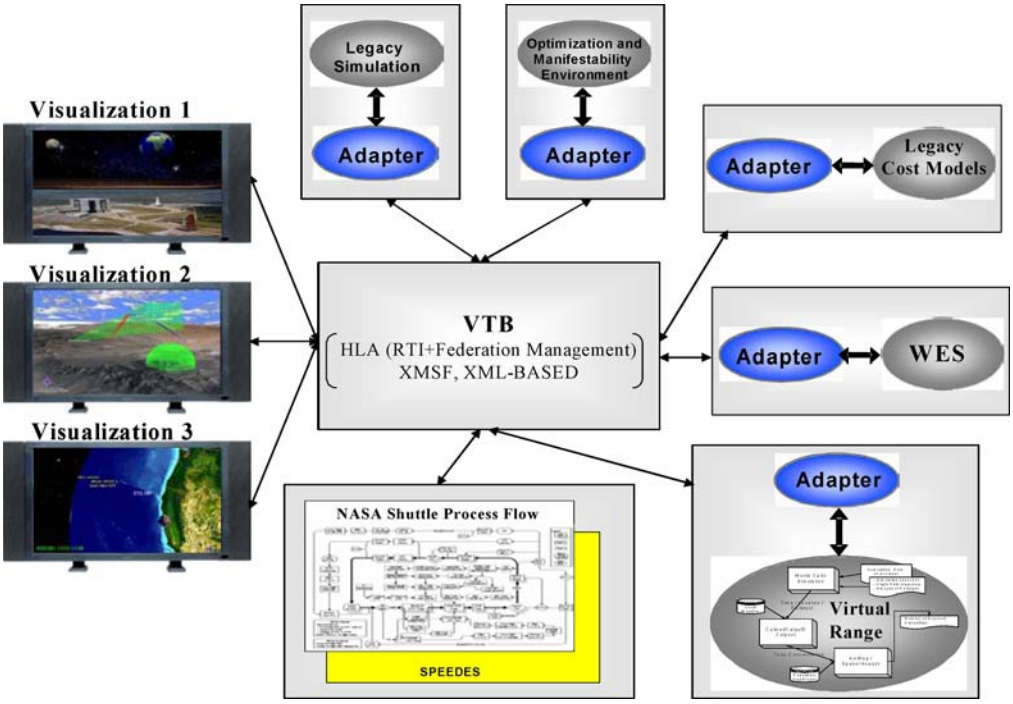

Figure 2 - Implementation of the VTB using the HLA

environment (Figure 1). The VTB provides an environment to integrate simulation models developed for specific elements of space operations into an interactive simulator network that supports a single view of operations. For instance, NASA KSC has models that have been developed over time by different sources. These existing models ("legacy" models) have been developed from different points of view and for different aspects of the operation cycle. They support different levels of resolution, and have selected different representation methods for internal entities, activities, and interactions (VTB Team 2003).

2.3 The VTB/HLA integration

Figure 2 depicts a conceptualization of the implementation and functionality of the VTB using the HLA. The VTB follows standards set by the DOD and the Institute of Electrical and Electronic Engineers (IEEE) for the integration of models. The High Level Architecture (HLA) is one of those standards. The VTB will follow HLA as the principal framework to integrate all the different types of models that need to be a part of the VTB. For example, a spaceport can be represented using different types of models using different information, spaceport size, and operation. The simulation system will be a subsystem that will evolve over time to meet this important requirement.

The VTB employs object models and object-oriented methods to exercise a hierarchical description of entities, activities, and interactions represented in the integrated models.

3 A VTB/HLA INTEGRATION EXAMPLE

Many factors contribute to a launch vehicle launching on time. The launch vehicle, spacecraft, and supporting range must all be ready to go at the desired launch time in order for the launch to occur. Each of these elements has supporting systems consisting of hundreds of subsystems and millions of individual components. Thousands of opportunities exist for technical system failure or human error. Some factors that can impact launch decisions, such as inclement weather and launch area intrusions, are out of the control of the launch officials. The different elements affecting launch decisions are addressed through two simulation models that were built independently.

The first model simulates Space Shuttle flow from processing at the Orbiter Processing Facility (OPF), through transport to the launch pad, liftoff, mission, landing at KSC, and refurbishing at OPF to get ready for a new launch. This is a simplified version of the conceptual flow diagram described by the Space Shuttle Processing Model (Cates, Steele, Mollaghasemi, and Rabadi 2000). A single shuttle is used to route between the different facilities and launch operations at KSC. All processing times come from the real-world data included in the Space Shuttle Processing Model.

In the Space Shuttle Processing Model, when the orbiter reaches the launch pad and is ready for launch, the simulation generates a random variable to determine the time that will elapse until the launch occurs. This time follows a theoretical distribution that closely matches events as historically observed. Those events account for historically observed instances of delays or scrubs that affected the launch process. A delay means the launch is postponed for a short time but still occurs on the expected date. A scrub means the launch is postponed for at least one day.

To illustrate the VTB capabilities and the procedure needed to combine existing computer simulations, the randomly generated delay (or scrub) in the Space Shuttle Processing Model was deleted, and processing requiring Shuttles to wait on the launch pad until an external authorization for launch is received was added.

To generate launch authorization commands, a second model independently simulates the range, the launch pad, and other spaceport facilities. This model focuses on events occurring in the range and in the processing facilities that can cause launch delays or launch scrubs due to mechanical or electrical failure.

Although both models discussed here were built using Arena™, either one of them (or both) could have been built using any other commercial simulation software that is available.

3.1 The (Modified) Space Shuttle Simulation Model

The (modified) Space Shuttle Simulation Model ("Model-1") is a mini model of space shuttle operations created in Arena™. Here, a single shuttle is used to move between the different facilities and launch operations at KSC. The Shuttle starts processing at the OPF (Orbiter Processing Facility). All processing times come from real-world data. After OPF processing, the shuttle is routed to the PAD where it completes PAD processing and sends a signal to Model-2 (the Launch Delay and Scrub model) that it is ready to launch. At this point, Model-1 waits for a GO/NOGO signal from Model-2.

Figure 3 shows the Shuttle waiting in the launch pad for authorization for liftoff. The dialog box in Figure 3 is displayed at the moment when the authorization signal is received.

As soon as Model-1 gets the signal from Model-2 to launch, it will route the shuttle to orbit, where it will finish the orbiting process. At the end of the orbiting process, the model checks for the end-of-mission day and lands the shuttle at KSC. After the shuttle lands at KSC, the shuttle's flight number is checked. If the flight number is 8, the shuttle is sent to Palmdale for maintenance. Otherwise it continues the cycle from the OPF. If it is sent to Palmdale, it finishes Palmdale processing and returns back to the OPF.

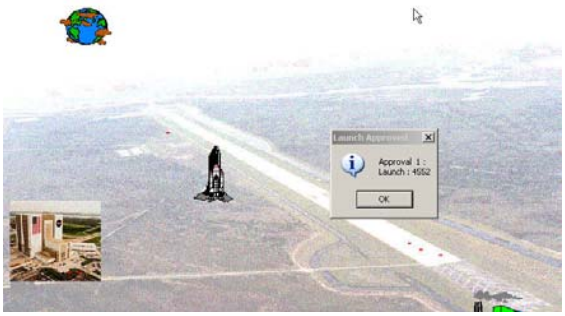

Figure 3 - The (modified) Space Shuttle Simulation Model

3.2 The Launch Delay and Scrub Model

The Launch Delay and Scrub Model ("Model-2") implements the scrub and delay logic using the historical scrub and delay probabilities for a shuttle launch. As soon as Model-2 gets the signal from Model-1, Model-2 generates a dummy entity to run the launch counter starting from 3 days. This entity enables the repeated evaluation of conditions that would call for a scrub for all times during the countdown phase. If there is a scrub, this entity is sent to the start of the sequence; i.e., it will restart the countdown counter.

If there is no scrub, the model checks for delays. The time and duration for a delay is generated randomly. If the entity encounters a delay, the launch is postponed for the duration of the delay. If the entity does not encounter any scrubs or delays, it runs the counter for 3 complete days, and at the last moment it sends the signal for launch *through the RTI* to Model-1. After passing the signal, the entity goes to the start of the sequence and waits for the signal for the next launch. This model also maintains information about the number of scrubs and the current delay time for the launch.

Figure 4 displays the Launch Delay and Scrub

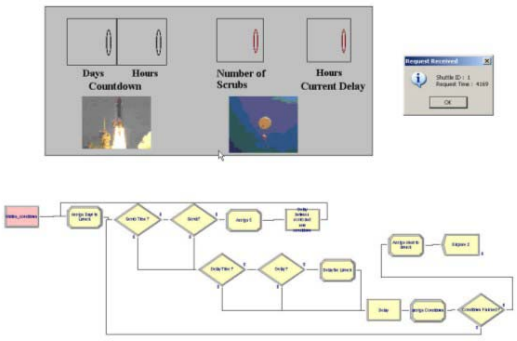

Figure 4 - The Launch Delay and Scrub Model

Model as it evaluates the weather and technical status of the range and launch facilities prior to authorizing the Shuttle for liftoff. The dialog box in Figure 4 shows the moment when the request for authorization message is received.

3.2.1 The Logic and Data behind the Launch Delay and Scrub Model

Historical information exists for the average number of system failures per month. A system failure is defined as a system or component failure that would result in a launch scrub. Launches can continue with many non-operational individual components or subsystems as long as a backup exists or the subsystem is not mission critical, safety critical, or has been designated as mandatory for this mission.

Many factors (see Table 1) affect the ability of launch vehicle to successfully launch on time. The launch vehicle, spacecraft, and supporting range must all be ready to go, simultaneously, in order for the launch to occur. Each of these elements has supporting systems consisting of hundreds of subsystems and millions of individual components. Thousands of opportunities exist for technical system failure or human error.

Table1 – Factors affecting delays and scrubs

System	Subsystem	Failure Rate
Launch Vehicle	Airborne Systems	1 failure per month
	Ground Systems	3 failures per month
Spacecraft	Airborne Systems	0.5 failures per month
	Ground Systems	2 failures per month
Range	Telemetry Systems	1 failure per month
	Tracking Systems	2 failures per month
	Command Systems	1 failure per month
Other factors	Weather	Lookup table – varies by month
	Launch Area clear	Lookup table – varies by month

Using the historical data, it was determined that for the launch vehicle, there was a 10.5% chance of the launch vehicle element causing a scrub. For the spacecraft, there was a 6.8% chance of causing a launch scrub. Other factors such as inclement weather (see Figure 5) and launch area intrusions (for example, a pleasure boat or an unauthorized aircraft entering a restricted area, see Figure 6) are out of the control of the launch officials.

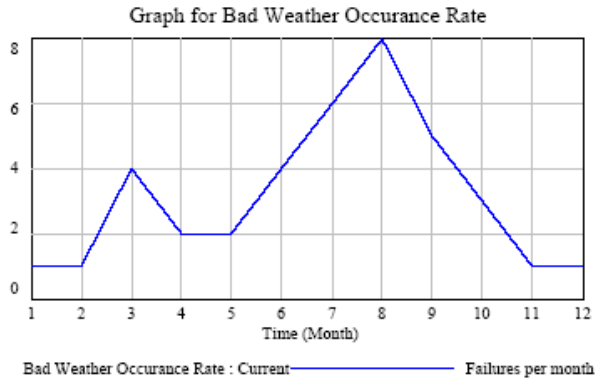

Figure 5 - Bad weather occurrence

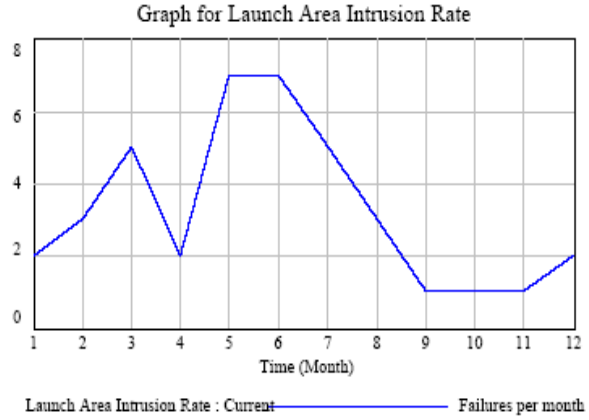

Figure 6 - Launch area intrusions

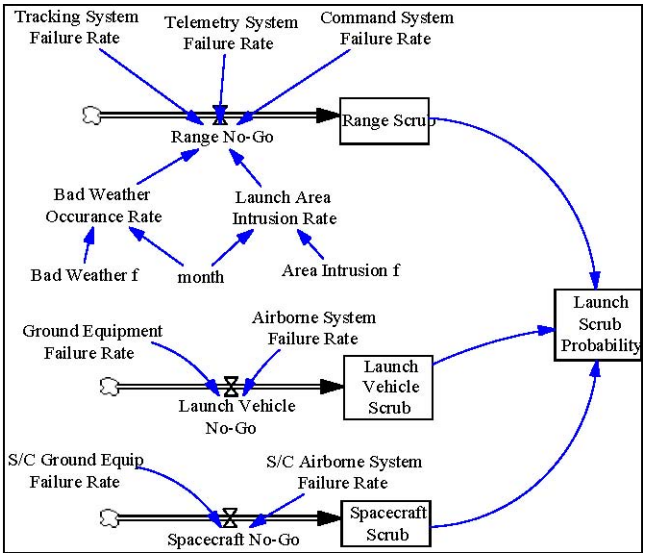

Figure 7 - Contributions to delays and scrubs

All of the hardware systems had a constant failure rate, except for two items, weather and launch area clearance that varied significantly with the time of year. In these cases, lookup tables were created to model the average "bad occurrence" per month for each month of the year.

A simplified model (see Figure 7) depicting the different contributions to delays and scrubs and their relationships was built using a System Dynamics approach.

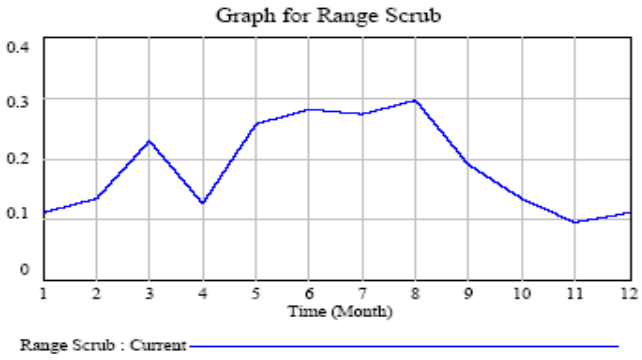

Figure 8 – Combined contribution of weather and range intrusions to range scrubs

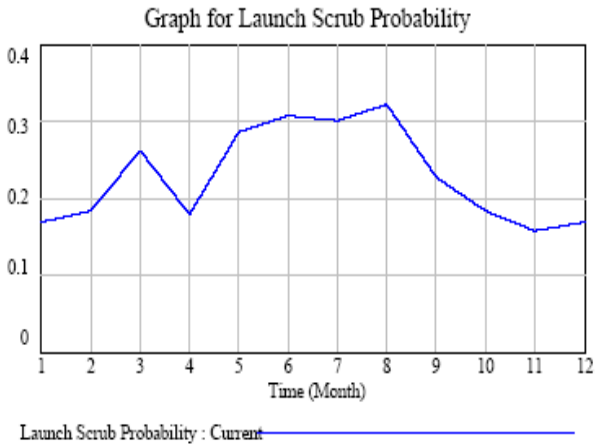

Figure 9 - Overall probability of a delay of scrub

The combined contribution of weather and range intrusions to range scrubs is depicted in Figure 8. The probability of a range scrub varied by month since the weather and launch area surveillance components also varied. It varied from 10% to 30% depending on the month, with the spring and summer months showing a higher chance for a scrub.

The overall launch scrub probability is shown in Figure 9 and varies between 16% and 32%, depending on the month. This data is useful for financial and schedule planning for launch vehicles.

3.3 VTB/HLA integration details

Figures 10 and 11 summarize the integration that occurs between the modified Space Shuttle Simulation Model (Model–1) and the Launch Delay and Scrub Model (Model-2). This integration and all messages exchanged occur over the RTI.

As illustrated in Figure 10, when one of the models requests it, the RTI creates a federation and lets the model join it. Later, when the second model joins the federation, the RTI synchronizes their clocks. Note that in most stand-alone simulation models, the simulation usually starts at time zero and advances in whatever time units the model is designed to use. When integrated using the RTI, the models are synchronized so that their corresponding clocks advance at the appropriate time for the combined distributed simulation. In this example, the models are synchronized around a specific calendar date and then progress in parallel while keeping their joint behavior synchronized. After both simulations run their respective courses, the models resign from the federation and, when the last one does so, the RTI removes the federation and closes down.

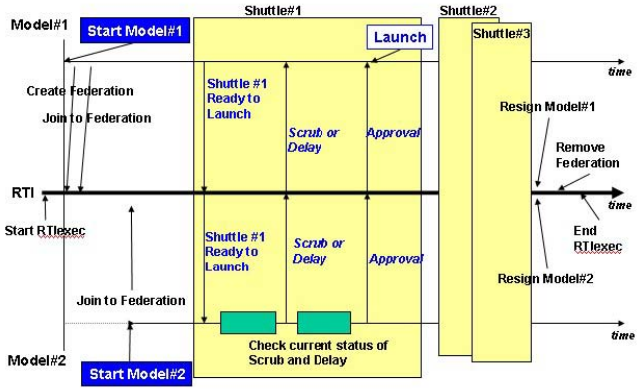

Figure 10 - Interaction between the (modified) Space Shuttle Simulation and Launch Delay and Scrub Simulation Models

Figure 11 shows the combined operation of the models depicted in Figure 3 and Figure 4 and illustrates the messages that are passed between the models.

3.3.1 The DMS Adapter

The integration of the VTB with the HLA is accomplished using the National Institute for Standards and Technology's (NIST) Distributed Manufacturing Simulation (DMS) Adapter. Figure 12 shows how the integration is done. The DMS Adapter is a component of an HLA-based infrastructure for distributed simulation of manufacturing facilities. The adapter was developed by NIST as part of the MISSION project, an

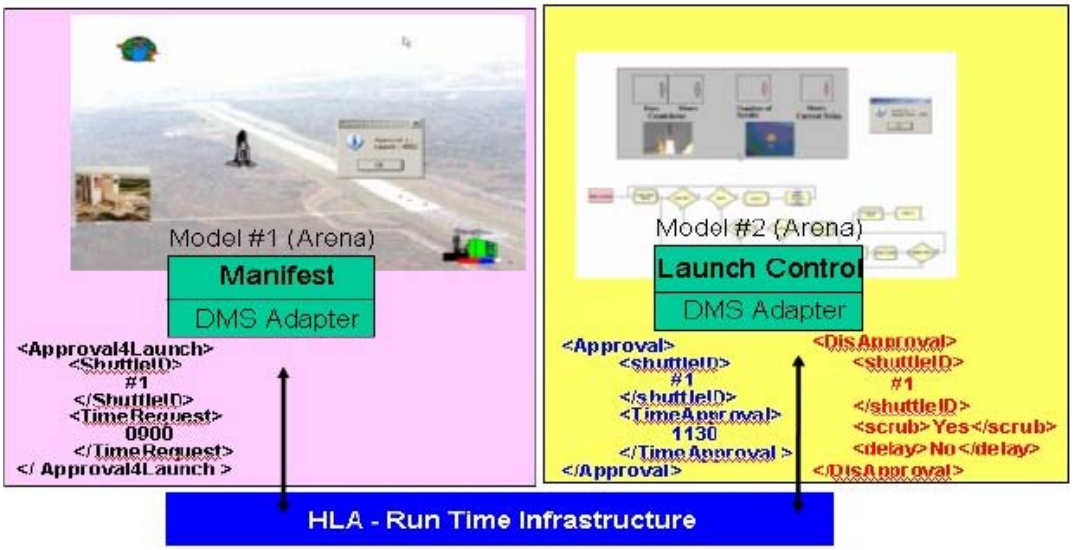

Figure 11 - distributed Shuttle Process Simulation using the DMS Adapter

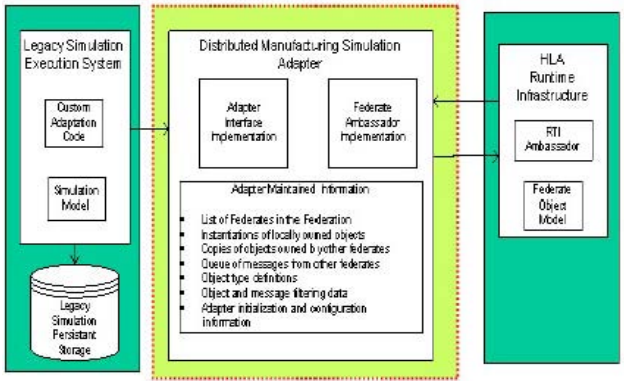

Figure 12 – The DMS Adapter

international, collaborative project, part of the international Intelligent Manufacturing Systems (IMS) Program (see www.ims.org).

The DMS Adapter's infrastructure was designed to support the integration of different manufacturing simulations with each other and with other manufacturing software applications. Applications that might be integrated using the DMS adapter include: new or existing simulation created with existing, non-HLA-compliant simulation development tools; existing enterprise software applications dealing with non-simulation situations (production planning, human resources, inventory control, supply chain information, finance and accounting, instruments data collection, etc); or general non-simulation and non-manufacturing oriented legacy software applications.

One of the goals of the DMS Adapter is to minimize the changes needed for simulations to participate in an integrated manufacturing simulation run. The DMS Adapter encapsulates the functionality of the HLA and exposes an integration architecture that provides similar functionality, but in a manner that is easier to use in a manufacturing environment. It provides mechanisms to coordinate the time between legacy simulations, facilitate message exchange, and provide facilities for object creation, update, storage, deletion, and transfer of ownership.

The DMS Adapter facilitates the adoption of distributed simulation in manufacturing environments by providing an interface that reduces the complexity of integrating simulations using HLA to a level that is practical for manufacturing simulations. It provides a simplified time management interface, automatic storage for local object instances, and simplified object and interaction filtering. It also eliminates the need to develop custom federate ambassador implementations for existing non-HLA simulations. If incorporated into each federate, the DMS Adapter works with the RTI to manage the exchange of object and interaction information between federates. A conceptual view of the structure of a simulation integrated with the DMS adapter to is shown in Figure 12.

3.3.2 The use of eXtensible Markup Language (XML) and the Unified Modeling Language (UML)

The data model used by the Adapter for the integration of the VTB simulations must incorporate a vocabulary that captures all the features that are shared between all NASA Shuttle processes. The vocabulary

must be common so that terminological differences between processes are reconciled and each feature is represented only once. Once a common vocabulary is constructed, each process analysis will only have to define a single interface instead of customized interfaces for each additional analysis. XML schemas are used to define XML messages that will be exchanged between the different simulations in the VTB using the DMS Adapter. Figure 13 shows an example of one of the messages that will be sent between the simulations.

```
<Launch Operation>
  <Countdown status = "Not Initiated" />
  <Launch Time>
    04/12/2004 17:00:00
  </Launch Time>
  <Pad>12</Pad>
  <Spacecraft Ground Station>
    <VehicleStatus Value="Normal" />
  </Spacecraft Ground Station>
  <Mission Director Center>
    <RangeSafetyStatus Value="Normal" />
    <EasternRangeStatus Value="Normal" />
    <Weather Value="Normal" />
  </Mission Director Center>
  <Spacecraft Mission Control Center >
    <SpacecraftNetworkStatus Value="Normal" />
  </Spacecraft Mission Control Center >
  <Range Operations Control Center />
</Launch Operation>
```

Figure 13 - XML Message Structure

To be able to define the XML messages to be exchanged the following approach was taken.

1. The messages for simulations that are a part of or potentially a part of the VTB are represented as Unified Modeling Language (UML) classes. UML class diagrams can then be used to analyze the potential for integrating new simulations in to the VTB, and to facilitate the development of new XML schemas that define new messages that can be supported by the simulations that are a currently a part of the VTB.
2. UML sequence diagrams can be used to design the protocols necessary to synchronize the current VTB simulations with other simulations that could potentially be a part of the VTB.

3.3.2. Future work

The main objective of the VTB team effort is to develop a new and unique collaborative computing environment where simulation models can be hosted and integrated in a seamless fashion. This collaborative computing environment will be used to build a "Virtual" Spaceport. The Virtual Range is a prototype of a virtual engineering environment to study the safety criteria of current and next-generation space vehicles during launch and range operations. The focus of our future work will be to integrate and develop the Virtual Test Bed with the Virtual Range. The emphasis is on the integration of the VTB operations and the Virtual Range models.

The modular architecture of the VTB enables the analysis of new vehicle types (e.g., the Crew Exploration Vehicle (CEV)) and the study of other launch sites. It is anticipated that the current environment will be extended to support the integration of other discrete-event simulations of KSC operations, and to make greater use of the High Level Architecture. These developments will be reported on in future papers.

REFERENCES

Bardina, Jorge. (2001). Intelligent Launch & Range Operations. Jorge Bardina. NASA ARC, 2001

Barth, T. (2002). Found in Space, IEE Solutions.

Brutzma, D., Zyda, M., Pullen, M., and Morse, K. L. (2002), "Extensible Modeling and Simulation Framework (XMSF). Challenges for Web-Based Modeling and Simulation".

Cates, G.R., Steele M.J., Mollaghasemi, M., Rabadi. G. (2002), Modeling The Space Shuttle, In Proceedings of the Winter Simulation Conference.

Cates, G.R., Steele M.J., Mollaghasemi, M., Rabadi. G., Sepulveda, J.A. (2002), Simulation, Modeling and Analysis of Space Shuttle Flight Hardware Processing, In Proceedings of the World Automation Congress.

Dahmann, J.S. Fujimoto, R.M. Weatherly, R.M. (1998), The DoD High Level Architecture: an update, in Winter Simulation Conference Proceedings.

Earth Tech. (1997). User Manuals for CALPUFF Version 5. Earth Tech. (2002). Addendum for CALPUFF Version 6.

Hibino, H. Fukuda, Y. Yura, Y. Mitsuyuki, K. Kaneda, K. (2002), Manufacturing adapter of distributed simulation systems using HLA, Winter Simulation Conference, Proceedings.

Intelligent Systems Project. (2000). Intelligent Systems Program Plan, NASA. November 7, 2000.

Kuhl, F., Riddick, F. (2002) Distributed Manufacturing Simulation Adapter.

Kuhl, F., Weaver, R, Dahmann, D. (1998). Creating Computer Simulation Systems, Prentice Hall. National Research Council. (2000). Streamlining

Space Launch Range Safety. National Academy Press.
Rabadi, G. (2001). KSC/UCF Shuttle Simulation Model, Retrieved August 2003 from www.amso.army.mil/ smart/conference/ 2001/18-apr/rabadi.ppt
Rajkumar, T., Bardina. J.E. (2003), Web-based Weather Expert System (WES) for Space Shuttle Launch, NASA Ames Research Center , Moffett Field, California.
Rechtin, E. (1991). Systems Architecting, Prentice Hall, 1991. Rechtin, E. and Maier, M. (1997).The Art of Systems Architecting, CRC Press, 1997. VTB Team. (2003). Model Assessment Report.

ACKNOWLEDGMENTS

We wish to thank Mr. Jeppie Compton from All Points Logistics, Mr. Cary Peaden from NASA Kennedy Space Center, and Dr. Jorge Bardina from NASA Ames Research Center for sharing their experience and knowledge with us as we worked. We are also grateful for all the good work provided by the graduate assistants in our Center for NASA Simulation Research Group not listed as coauthors: Fred Gruber, Mario Marin, Oscar Martínez, Karthik Nayaranan, Amith Paruchuri, Deepak Rachapalli, Amani Saleh, David Sepúlveda, and Amit Wasadikar.

Mention of commercial products in this paper does not imply approval or endorsement of any commercial product by NIST. Parts of this project are funded by NIST's SIMA Program. SIMA supports NIST projects applying information technologies and standards-based approaches to manufacturing software integration problems. The work on the DMS Adapter described in this paper was funded by the United States Government and is not subject to copyright.

AUTHOR BIOGRAPHIES

JOSÉ A. SEPÚLVEDA, Ph.D., P.E., is an Associate Professor in the Department of Industrial Engineering and Management Systems at the University of Central Florida in Orlando, Florida. He received an Ingeniero Civil Químico degree from the Universidad Santa María, Valparaíso, Chile, and MSIE, MPH, and Ph.D. (Industrial Engineering) degrees from the University of Pittsburgh. Dr. Sepúlveda is a registered Professional Engineer in Florida and Chile. Dr. Sepúlveda's major areas of research interest are object-oriented simulation, simulation optimization, risk analysis, catastrophe response, measuring and modeling training effectiveness, tasks scheduling in complex and risky environments, and applications of industrial engineering and simulation in health care. He has written two books and numerous publications. His e-mail address is sepulved@mail.ucf.edu.

LUIS RABELO, Ph.D., is an Associate Professor in the Department of Industrial Engineering and Management Systems at the University of Central Florida in Orlando, Florida. He received dual degrees in Electrical and Mechanical Engineering from the Technological University of Panama and Master degrees from the Florida Institute of Technology (Computer Engineering, 1987) and the University of Missouri-Rolla (Engineering Management, 1988). He received a Ph.D. in Engineering Management from the University of Missouri-Rolla in 1990 where he also did Post-Doctoral work in Nuclear Engineering and Artificial Intelligence in 1990-1991. He also holds dual MS degrees in Aerospace Systems Engineering & Management from the Massachusetts Institute of Technology (2001). He has over 125 publications and three international patents. His experience includes Ohio University, BF Goodrich Aerospace, Honeywell Laboratories, the National Institute of Standards and Technology (NIST), NASA, and the Massachusetts Institute of Technology. Dr. Rabelo has expertise in simulation modeling, aerospace engineering, software engineering, and complex systems. His e-mail address is lrabelo@mail.ucf.edu.

JAEBOK PARK is currently a Ph.D. candidate in the Institute for Simulation & Training at the University of Central Florida in Orlando, Florida. He received a B.S. in Electrical Engineering from Korea Naval Academy in 1988, and a M.S. in Computer Science from the Western Illinois University in 1994. His Research interests include Distributed Simulation, High Level Architecture and Visualization for HLA-based distributed simulations. His e-mail address is ja969179@pegasus.cc.ucf.edu.

FRANK RIDDICK works with the Manufacturing Simulation and Modeling Group currently at the National Institute of Standards and Technology (NIST) in Gaithersburg, MD. He holds a MS degree in Applied Mathematics from Purdue University. Frank has worked in developing and promoting the use of simulation related standards in the Object Management Group and the Simulation Integration Standards Organization. His e-mail address is riddick@cme.nist.gov.