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

PARALLEL DISCRETE EVENT SIMULATION OF SPACE SHUTTLE OPERATIONS

Luis Rabelo
Jose Sepulveda
Mario Marin
Amith Paruchuri
Amit Wasadikar
Karthik Nayaranan

Department of Industrial Engineering and Management Systems
4000 Central Florida Blvd.
University of Central Florida
Orlando, FL 32816, U.S.A.

ABSTRACT

This paper describes the application of parallel simulation techniques to represent structured functional parallelism present within the Space Shuttle Operations Flow, utilizing the Synchronous Parallel Environment for Emulation and Discrete-Event Simulation (SPEEDES), an object-oriented multicomputing architecture. SPEEDES is a unified parallel simulation environment, which allocates events over multiple processors to get simulation speed up. Its optimistic processing capability minimizes simulation lag time behind wall clock time, or multiples of real-time. SPEEDES accommodates increases in processes complexity with additional parallel computing nodes to allow sharing of processing loads. This papers focuses on the whole process of translating a model of Space Shuttle Operations Flow represented in a process-driven approach to object oriented design, verification, validation, and implementation.

1 INTRODUCTION

NASA implemented the Intelligent Launch and Range Operations (ILRO) Program at Ames Research Center (ARC) to perform initial studies of a test bed with a demonstration (Bardina 2000, Intelligent Systems Project 2000). An evolution of the ILRO test bed is the Virtual Test Bed (VTB) Project. The objective of the VTB Project is to provide a collaborative computing environment to support simulation scenarios, reuse, and integration of multidisciplinary models that represent elements of operations, the range, and spaceports. The VTB will provide several benefits, such as a risk management, evaluation of legacy and new vehicle framework, a technology pipeline, and a knowledge management enabler. The VTB will leverage current technological developments from NASA ARC in intelligent databases to present data and results as usable knowledge with associated security constraints and human-centered computing (HCC).

Virtual test bed environments are the next S-curve of the computer-aided design (CAD) systems. These "advanced" CAD environments will not only combine the 3D solid modeling capabilities of the current CAD systems, but also will integrate, in a seamless fashion, models that represent the different stages of the lifecycle of a system. Therefore, the 3D solid models of the mechanical/physical design will be integrated with the materials models, dynamics models, environmental models, operational models (including operators, crew members), safety models, and other type of models. These virtual environments will go beyond the virtual world attempts of the 90's. The virtual worlds of the 90's concentrated on the look and feel dimensions. However, a virtual test bed environment will go beyond the former cosmetic approach with higher realism and fidelity and with more emphasis on engineering. Virtual test beds will create multi-disciplinary and collaborative design spaces.

From another viewpoint, the current CAD environments concentrate on single dimensional views of a system but are still not adequate for complex systems design. One interesting characteristic of a complex system is that it is by default a system of systems. To be faithful to concurrent engineering principles, you have to study the interactions among the different systems that are elements of the complex systems. This system of systems is non-linear in nature and the interactions among the different components bring interesting emergent properties that are very difficult to visualize and/or study by using the traditional approach of decomposition.

According to Barth 2002, "Spaceport technologies must employ a lifecycle 'system of systems' concept in which major spaceport systems – launch vehicle processing systems, payload processing systems, landing and recovery systems, and range systems – are designed concurrently with the flight vehicle systems and flight crew systems." Therefore, the goal is to develop a VTB that can host the different models that represent the different systems and elements of a spaceport. These models on the VTB will work together in an integrated fashion, synthesizing into a holistic view and becoming a Virtual Spaceport. This Virtual Spaceport can be utilized to test new technologies, new operational processes, and the impact of new space vehicles on the spaceport infrastructure, supply chain, and the introduction of higher schemes of decision-making. A Virtual Spaceport will allow an intelligent visualization of the entire spaceport concept and the implementation of knowledge management strategies. The central goal of the VTB project is to provide a virtual environment of the launch and range operations at Kennedy Space Center (KSC). The VTB will integrate and adapt some of the existing simulation models and complement some of the gaps to create a unique mission environment for the ILRO program. This realistic NASA mission environment will provide scientists within the Intelligent Systems (IS) project from NASA with a computing environment where they can implement schemes for high-performance human-automation systems. This integration will require the development of a computer architecture that allows the integration of the different models and simulation environments using two modalities:

- Tightly-Coupled Integration
- Loosely-Coupled Integration

The computing infrastructure is implementing advanced ideas of integration, distributed and/or parallel computing, security, and Web-based technologies. Federation management is being developed. These federation management capabilities will allow other platforms to be integrated. However, these independent platforms will be integrated using a loosely coupled fashion. We believe that to successfully complete this task will require the development of an adapter to accomplish and/or facilitate this integration.

This computing infrastructure can be used to target applications in the design, development, and deployment of future-generation mission operations systems for the International Space Station (ISS). In addition, the Human-Systems Modeling (HSM) groups of the HCC project have been conducting research and development in computational modeling of distributed groups interacting with mission critical hardware/software elements. These interactions have included cognitive, physical, and social factors. In addition, there are developments in multi-person performance and work process modeling from the Multimodal Interface (MI) groups. These results will be implemented/integrated in the computing platform provided by the VTB.

2 SYSTEM ARCHITECTING ACTIVITIES

It is very well known that systems architecting integrates systems theory and systems engineering with architecting theory and the practice of architecting (Rechtin 1991; Rechtin and Maier, 1997). Conceptualization is the keyword for architecting. System conceptualization involves creativity and the recognition of potential users and perceived needs. System architectures are driven by the function, instead of the form, of the system. Systems Engineering provides the form. We utilized Quality Function Deployment (QFD) in this project and, in particular, a modified house of quality to guide our architecting activities.

QFD can be used to improve the process of introducing new ideas that translates the user requirements from concept to development and beyond. In the application of QFD, the initial phase involves the creation of a matrix called the House-of-Quality matrix due to its roof-like format as shown in Figure 1. The House-of-Quality used here follows modifications of the matrix of change (MOC - <http://ccs/mit.edu/moc>). This House-of-Quality has a list of the user needs/benefits (organized on the left side of the House). This list of needs/benefits is translated into the features (technical) needed (design requirements of a solution to satisfy the needs and/or provide the benefits).

The potential users "verbalized" their needs and the benefits desired. These "verbalized" needs and benefits were clustered into six areas:

- To decrease cycle time during the evaluation of vehicles and systems compliance to safety criteria.
- To improve the evaluation of vehicles and systems compliance to safety criteria.
- To synthesize in a single view the different elements of the Space Transportation System.
- To develop precautions/emergency response measures to support range safety and security in a systematic manner.
- To accelerate the introduction of new technologies (technology pipeline) and new operational procedures and help configure the current ones.
- To establish a repository of knowledge and Knowledge Management strategies.

The next step was to translate these needs/benefits to the desired design requirements/technical features of the VTB. The identification of the different features was the product of discussions. The ranking of the technical features is providing a guideline for the selection of a viable computing architecture for the VTB project. We understand that these technical features are very ambitious to ob-

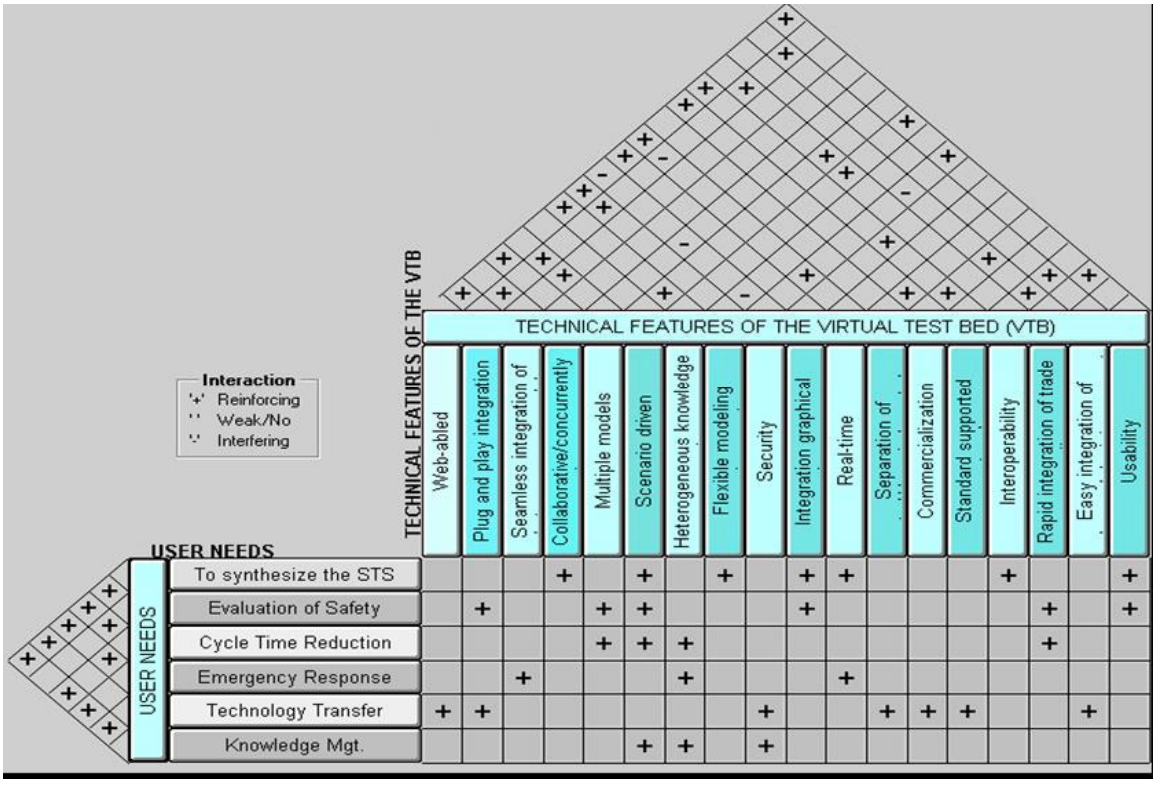

Figure 1: Interactions among the Different User Needs and the Technical Features of the VTB

tain and some of them require trade-offs to be implemented. The following is a listing of the technical features:

- Web-able: This will guarantee the distributed nature and user acceptance required for the different applications of the VTB.
- Plug-and-Play Integration: The models should be able to be integrated without complications.
- Seamless Integration of Different Types of Models: A spaceport can only be represented using different types of models (sizes and nature).
- Collaborative/Concurrent Environment: The computing environment should allow the implementation of collaborative design spaces (not limited by geographical constraints).
- Multi-Model: To represent the range and operations of a spaceport, multiple models describing the different elements and processes will be integrated. In addition, scalability is an important design requirement.
- Scenario Driven: Scenarios representing impediments or lessons learned should be able to be executed and archived.
- Extensive/Heterogeneous Knowledge Repository: Knowledge integration of different forms is expected.
- Flexible Modeling Environment: Ontologies could be used to provide the semantic primitives for a modeling language.
- Real-Time Environment: There is a need in launch operations for real-time simulation and decision-making.
- Security Layers: Security has to be implemented in different manners and layers to provide the required level.
- Integrated Graphical Environment: The user should be able to see the output in different ways and media. The environment can be integrated with Visual Query Languages (VQLs).
- Separation of Control/Application Logic/Architecture/Domain Knowledge: The VTB architecture should be independent of the domain knowledge.
- Commercialization (Business Development): One of the goals of NASA is to commercialize the technologies developed.
- Standard Supported Protocols: Standardization is an important goal in the design.
- Interoperability/Cross Platform Capabilities: Metadata mappings among different formats, common representations, reusability, and cross platform capabilities.
- Rapid Integration of Trade Studies: The rapid integration of trade studies for road mapping

and technology transfer activities is a strong design requirement.

- Easy Integration of Advanced Decision Support Tools: One of the objectives of the VTB project is to advance the decision-making capabilities for range and operations (e.g., more risk management and less risk avoidance).
- Usability (and User Friendliness): Human-centered computing principles taking into consideration the physical and cognitive models of the users must be implemented.

3 VTB ARCHITECTURE

The VTB Architecture is composed of the Integration User Interface, the Decision-Maker User Interface, the Security Component, the Integration System, the Simulation System, the Model Functions Manager, the Model Library Manager, the Database System, and the File Storage System (Figure 2). The Integration User Interface provides the capability to transfer a model to the VTB. The user can integrate an existing model (and create extensions to it) using the different tools and methodologies provided by this interface. The Decision-Maker User Interface is the simulation interface. The Decision-Maker can develop scenarios with the existing integrated models hosted on the VTB. The Security Component provides different levels of computer security such as password schemes, authentication, firewalls, Secure Socket Layer (SSL) implementations, maintenance and prevention mechanisms , certificates, and encryption. The Integration System takes the representation and user interface supports the execution and they develop the information outlined by the user (using the Integration User Interface) and formulate a hierarchical description of entities, activities, and interactions that is represented in an integrated model. The Simulation System executes the integrated model(s) according to the scenarios submitted by the user using the Decision-Maker User Interface. The Simulation System invokes the integrated model(s) from the VTB Host and the model's operation functions from the Model Functions Manager. The Model Functions Manager provides the business logic for the different transactions to save the different model configurations as specified by the Integration System. The Model Functions Manager also retrieves from the Database System and the File Storage System the simulation models, data, and configuration parameters needed by the Simulation System.

The Model Library Manager will support the development and management (retrieval, saving, configuration management) of the libraries. The Database System will store the model and its details. Finally, the File Storage

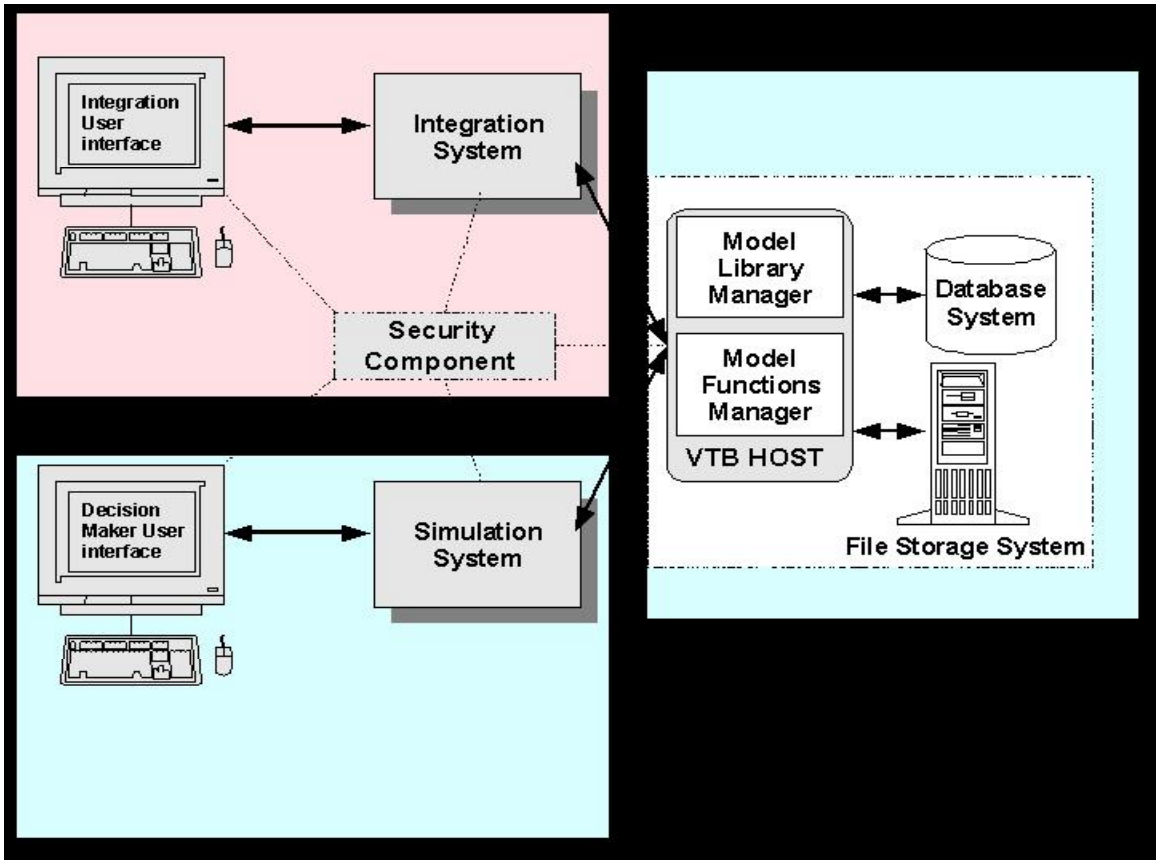

Figure 2: Basic Computer Architecture of the Virtual Test Bed (VTB)

System stores the model and its details in a scheme appropriate for facilitating the operations of the Simulation System and the interface with NASA Ames Research Center ILRO VTB transactions to save the different model configurations as specified by the Integration System. The Model Functions Manager also retrieves from the Database System and the File Storage System the simulation models, data, and configuration parameters needed by the Simulation System.

The Simulation System is one of the most sophisticated technological components of this architecture (Figure 3). The Simulation System provides an environment to execute integrated simulators/models developed for specific elements of space operations into interactive simulator networks to support a single view of operations. For instance, NASA KSC has existing models that have been developed over time by different sources.

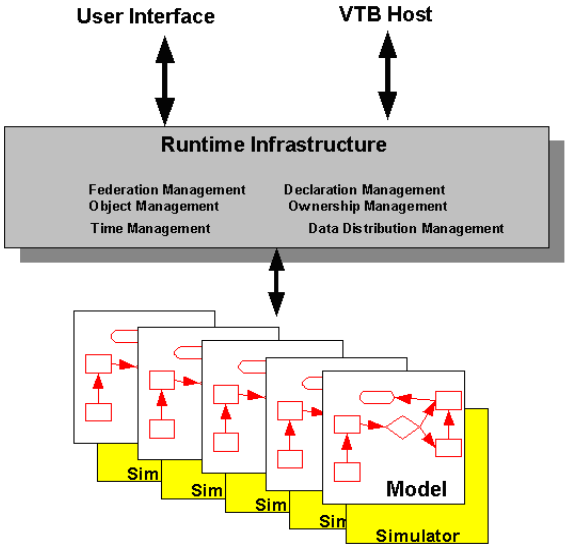

Figure 3: Conceptualization of the Simulation System using Federations (as Emphasized by the High-Level Architecture Dahmann et al., 1998, Dahmann and Morse, 1998, Dahmann 1999)

A spaceport can only be represented using different types of models (sizes and nature). The natures of the models that the Simulation System will execute in an integrated fashion are (1) discrete-event models, (2) continuous models (based on differential equations), (3) legacy simulation models (e.g., models developed in some limited COTS tools such as ARENA from Rockwell Software), and (4) static models (e.g., spreadsheets). The first version of the Simulation System has focused on discrete-event simulation. This first version has demonstrated the capabilities of hosting a discrete-event model that represents a high-level representation of the NASA Shuttle operations.

4 SYNCHRONOUS PARALLEL ENVIRONMENT FOR EMULATION AND DISCRETE-EVENT SIMULATION (SPEEDES)

The VTB team set out by searching for a general-purpose discrete-event simulator for the Simulation System. This simulator needed to simulate a wide variety of situations. The team identified some basic requirements for this simulator:

- The simulator needs to support different hardware architectures ranging from a distributed network of fast workstations to a single computer.
- C++, C, or Java need to be the main languages to support the development and implementation of the simulator.
- The simulator must provide interfaces for developing external interactions and federations (such as those supported by HLA). Application Program Interfaces (APIs) are important for this.
- The simulator must have capabilities to run geographically distributed simulations.
- The emphasis of the simulator structure needs to be object-oriented.
- The simulator should be capable of providing a framework to allow scripting languages to be written as command messages on top of the simulator layered-architecture.

We have been using SPEEDES as one of the discrete-event simulators for our HLA implementation because it meets the requirements mentioned. SPEEDES is a software framework/toolbox for building parallel C++ simulations. It is based on NASA-patented algorithms and it has good documentation. SPEEDES allocates events over multiple processors to get simulation speed-up (www.speedes.com). This feature enhances runtime, especially when exploiting the very large number of processors and the high-speed internal communications found in high performance computing platforms. SPEEDES is HLA compliant, and it provides two ways to connect to HLA:

- Built-in HLA Gateway
- Customized HLA Gateways

Another important characteristic of SPEEDES is object-orientation. SPEEDES' object-oriented architecture has a significant impact on the development of simulations. Individual classes can represent entities in a system. Such a representation, in turn, facilitates the distribution of the simulation models on different processors and the design of parallel simulation experiments. In addition, SPEEDES supports distributed simulation over the World Wide Web. This provides a very important advantage – a key feature of the World Wide Web for running a distributed simulation is the transparency of network heterogeneity, where interop-

erability of different networks is achieved through well-defined, standardized protocols such as HTTP and CGI. As a distributed discrete event simulation framework, it allows distribution of various objects over multiple processors and coordinates the simulation activities among various objects that are distributed. SPEEDES provides interfaces for developing external modules. These modules provides functionalities that allow interoperability between various simulation systems and tools that will make sure the globally distributed simulation executes efficiently. External modules connected to SPEEDES simulation control time advance of the simulation, receive information about the simulation state, and invoke events in the simulation. External modules can also be used to display simulation status and provide inputs through hardware to control the simulation. SPEEDES provides an advanced feature called Load Balancing. This feature enables the user to balance the objects that require more processing on a faster processor, leading to improvements in run time performance.

The distribution of objects in SPEEDES simulation can be done in two ways: (1) Automatic Object Placement and (2) Manual Object Placement. Two built-in algorithms called SCATTER and BLOCK do automatic Object Placement. SCATTER distributes objects like distributing cards in a card game. BLOCK distributes objects evenly across the processors. SPEEDES is an optimistic framework; any changes made to the state of the simulation object need to be restored in case an event is rolled back. SPEEDES also supports the ability to roll an event forward without requiring large amounts of memory overhead if the state that the event depends upon has not changed. This is known as lazy event re-evaluation.

5 PROCESS FLOW OF THE NASA SHUTTLE TRANSPORTATION SYSTEM

SPEEDES was utilized to simulate the process (at a high level) of the STS (Figure 4). The source of the process flow (high-level) model was the NASA Shuttle Simulation Model. The NASA Shuttle Simulation Model is a simulation model for the operational lifecycle of the Space Shuttle flight hardware elements through their respective ground facilities at KSC developed by NASA (Rabadi 2001) and the University of Central Florida (). The COTS tool used was Arena from Rockwell Software.

The modeling approach of the NASA Shuttle Simulation Model was done at a macro level. It included among others, the following Space Shuttle elements:

- The orbiter
- The main engines
- The left and right orbiter maneuvering system pods
- The forward reaction control system
- The solid rocket boosters
- The external tank.

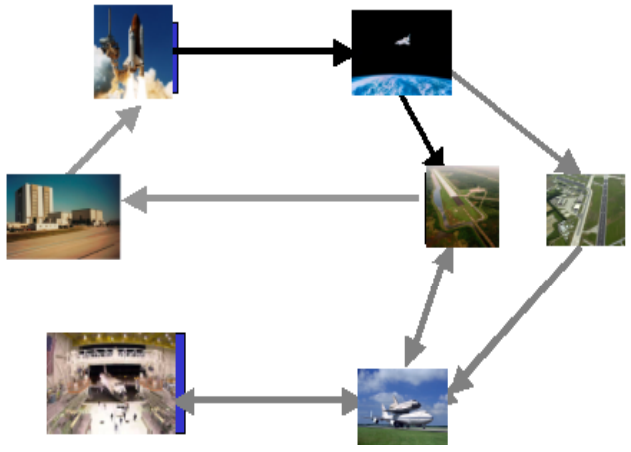

Figure 4: Operational Lifecycle of the Space Shuttle

The VTB team used an incremental approach to transfer the process flows from the COTS tool to SPEEDES. The Unified Modeling Language (UML) was used to develop the hierarchy of objects and some partial generation of the C++ code. These diagrams (Figure 4) guide in the definition of the object classes to be used throughout.

Defining the major object classes helps to narrow down the most relevant processes and events that take place. Other tasks or processes which are common to several classes can be differentiated as attributes or functions for the given objects.

The distributed simulation capabilities of SPEEDES were tested by doing different experiments by distributing the 41 objects of the STS Process Flow in different numbers of computers. These computers were even located in different geographical locations. For example, the SPEEDES server was installed at the University of Central Florida (Orlando, Florida) and two simulation nodes in two different computers at NASA ARC (Ames, California). The interactions among the different objects (in the different computers) produced a simulation that produced the original results using a single computer. The gains are in speed and the utilization of unique resources attached to each node. The environment will prove even more useful with the future additions to the original simulation model. These additions will allow for different resolution levels and the study of safety and human-behavior modeling issues.

5.1 Speedes Implementation

5.1.1 Stage I

The "NASA space shuttle model" previously developed in Arena was broken down into smaller modules to better understand the behavior of the various entities and processing objects. Distinguishing the objects to be modeled was a major task. Breaking the Arena model into smaller modules (Figure 5) helped in classifying the objects required to

form a infrastructure for developing the entire "NASA space shuttle model" in SPEEDES.

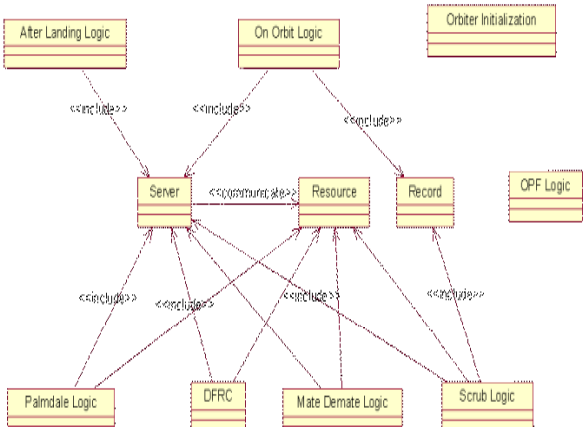

Figure 5: Class Diagram of Space Shuttle Processes

There is one entity (shuttle) and ten processing objects, which were reviewed and implemented, these objects are:

- Orbiter Processing Facility (OPF)
- launch pad (launch)
- on orbit (orbit)
- Kennedy Space Center (KSC)
- mate de-mate (MDM)
- global
- route
- Palmdale
- Shuttle
- Vehicle Assembly Building (VAB)
- Dryden Flight Research Center (DFRC).

The entire model developed in Stage I was named as "version 1." It consisted of only one hardware element, the orbiter, which would only go through the major processes that take place on a shuttle as it completes a flight. Each processing object contains many processes/servers. Stage I was the base needed to develop the platform for the future complexity and sub processes that needed to be modeled.

Executing version 1 on a single processor and multiple processors was a major accomplishment, reached early on. The results were identical with that of the Arena model. This proved that SPEEDES models provide consistent results whether running on one or more than one processor.

5.1.2 Stage II

Stage II was more focused on the sub-processes involved in each processing object that was modeled in Stage I and scaling the model with more hardware elements/entities and processing objects. More objects were then classified by distinguishing the entities and the processing objects.

Although the new entities were never modeled as a separate class, there were instances of the same class "shuttle"; for every entity/instance there was a separate set of state variables assigned. For example, the engine of the orbiter was assigned a primary variable called "engine." This variable concept was duplicated from the Arena model. Whenever a new entity was to be created, a new instance of the class shuttle was declared with different sets of state variables.

5.1.3 Stage III

SPEEDES can distribute or parallelize simulations to computers across networks such as LAN, WAN, or even the Internet. The processing speed of these nodes can be very different, which makes synchronization important. It could happen that a particular value needed by a process A is generated by another process B. So it is important that process A requests the value on or before the time process B generates it. This cannot be guaranteed, since the various nodes are computers with different configurations and hardware. So to ensure this the server uses two types of events: Rollback and Commit. Rollback is done when the data requested by process A has already been generated by B. So the server now asks process B to Rollback to the time where it had generated the value needed by A. Rollback ensures that process B generates the same values from the Rollback to the time before it did the Rollback. This makes sure that other processes which are dependent on B are not affected. Commit is done when all the processes dependent on B are in a state in which they do not require any value from process B, say after a point t in time. So the server will do a Commit at point t. Once a Commit is given, the system cannot Rollback to a point beyond t. Hence, to make a our process Rollback-able, the events must also be Rollback-able, i.e. the occurrences of say event E should have the same distribution before and after the Rollback. Therefore, we added several random generators Rollback-able to SPEEDES. The algorithms for the Gamma, Lognormal,, Weibull, Triangular, Continuous, Discrete, and Johnson were written in C++ and were validated by fitting the output curve of these random number generators in Arena's input analyzer tool. This was a very important addition to enhance SPEEDES.

5.1.4 Stage IV

The addition of more classes to this environment has been planned and will include the different visualization environments. Visualization is a very important feature of modern simulation modeling environments. The VTB team has been investigating different visualization paradigms. There are many visualization tools available. However, for space operations among the most sophisticated tools are the Real-Time Graphics Engine (RAGE) from White

Sands Missile Range, EDGE from Boeing Autometric, and customized environments using JAVA 3D and the Virtual Reality Modeling Language (VRML) and other extensions using the eXtensible Markup Language (XML), such as X3D, Web3D, and Xj3D. Figure 6 shows the development of multiple windows (one for each Shuttle) using JAVA 3D and VRML Objects and manipulated from different computers using SPEEDES. Figure 7 show a demo using Edge (from Boeing Corporation) for STS 107.

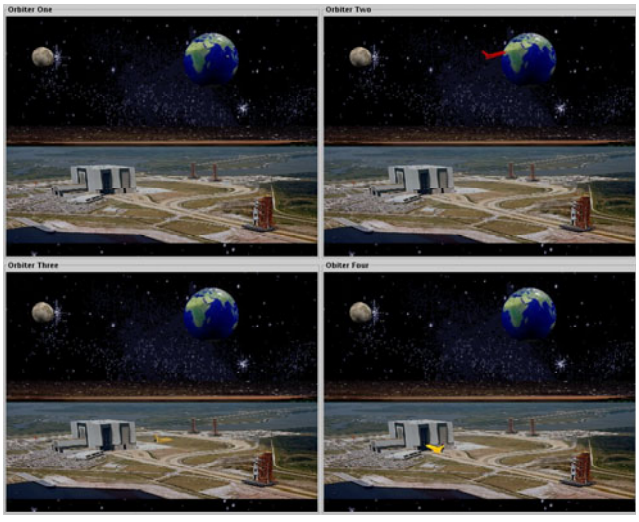

Figure 6: JAVA 3D and VRML Objects Distributed over the Web for SPEEDES

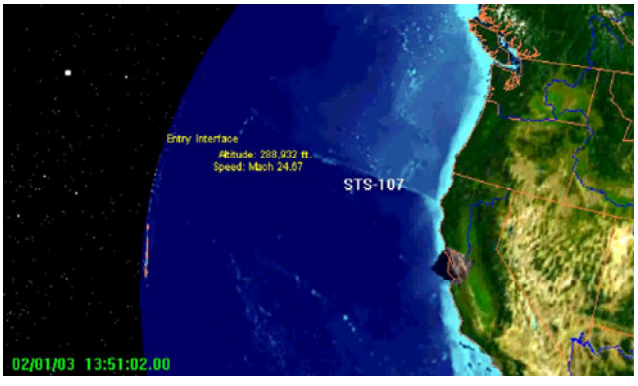

Figure 7: Visualization of STS 107 (Shuttle Columbia Feb. 1, 2004) using Edge. Edge is a GIS that Provides the Coordinates of the Earth and the Solar System.

6 CONCLUSIONS

Modeling of Space Shuttle Operations Flow represented in a process-driven approach has been demosntrated with the use of SPEEDES. All objects that make up the main physical systems and processes have been created with the help of UML. This acomplishment represents the starting point for the development of the entire spaceport operational architedture through object-oriented design.

The VTB has been designed as an architecture to facilitate the integrated execution of different simulation programs with other (non-simulation) legacy software. The architecture must deal with issues related to the coordination of different hardware platforms and components and different software components. The architecture must in addition 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.

We are using standards, distributed environments, and emphasizing open source schemes whenever possible. With decreasing budgets, spending too much for unique proprietary technology is no longer a sustainable option for NASA. NASA can benefit most by using Web-wide standards and industry best practices (learned and adapted from the eXtensible Model Simulation Framework (XMSF) experience (Brutzma et al., 2002).

REFERENCES

Bardina, Jorge. 2001. Intelligent Launch & Range Operation. NASA Ames Research Center, Ames, California.

Barth, T. 2002. Found in Space, IIE Solutions, April issue.

Brutzma, D., Zyda, M., Pullen, M., and Morse, K. L. 2002, Extensible Modeling and Simulation Framework (XMSF). Challenges for Web-Based Modeling and Simulation, Technical Opportunities Workshop Whitepaper, The MOVES Institute, Naval Postgraduate School, June 14.

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

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

Dahmann, J.S. Morse, K.L. 1998, High Level Architecture for simulation: an update.

Dahmann, J.S. 1999, The High Level Architecture and beyond: technology challenges, in Thirteenth Workshop on Parallel and Distributed Simulation, Proceedings.

Rabadi, G. 2001. KSC/UCF Shuttle Simulation Model, Retrieved August 2003 from: <http://www.amso.army.mil/smart/conference/2001/18-apr/rabadi.ppt>.

Rechtin, E. and Maier, M. 1997.The Art of Systems Architecting, CRC Press: Boca Raton, Florida.

Rechtin, E. 1991. Systems Architecting, Prentice Hall: Englewood Cliffs, New Jersey.

SPEEDES, Users Guide 2003, Metron Corporation, San Diego, California. Retrieved May 1, 2003, from: <http://www.speedes.com/docs/SpUG.pdf>.

SPEEDES API Reference Manual 2002. Retrieved May 1, 2003, from <http://www.speedes.com/docs/APIReference.pdf>.

AUTHOR BIOGRAPHIES

LUIS RABELO is an Associate Professor in Industrial Engineering and Management Systems at the University of Central Florida, Orlando, USA. He received his Ph.D. in Engineering Management from the University of Missouri at Rolla in 1990. He has also received graduate degrees in Electrical Engineering, Engineering Management, Systems Engineering, and Management from the Florida Institute of Technology, the University of Missouri-Rolla, and the Massachusetts Institute of Technology. He has worked as a scientist and technology leader for BFGoodrich Aerospace, Honeywell, Ohio University, and the National Institute of Standards and Technology. His research interests are in control theory, simulation, artificial intelligence, and supply chain management. He is a member of IIE. His email is <lrabelo@mail.ucf.edu>.

JOSE SEPULVEDA is an Associate Professor in Industrial Engineering and Management Systems at the University of Central Florida, Orlando, USA. He received his Ph.D. in Engineering Management from the University of Pittsburgh in 1981. He has also received graduate degrees in Public Health and Industrial Engineering from the University of Pittsburgh, and the degree of Ingeniero Civil Químico, from Universidad Santa María, Valparaíso, in Chile. His email is <sepulved@mail.ucf.edu>.

MARIO MARIN received his Masters in Simulation Modeling and Analysis from the University of Central Florida. He is currently pursuing a Ph.D. in Industrial Engineering at the University of Central Florida. His research interests are mainly simulation integration of operations and HLA and XML technologies. His email is <mmarin@mail.ucf.edu>.

AMITH PARUCHURI received his Masters in Simulation Modeling and Analysis from the Institute of Simulation and Training (IST) in Orlando Florida. His research interests are mainly in visualization, 3-D graphics, and distributed simulation. His email is <amithparuchuri@yahoo.com>.

AMIT WASADIKAR will receive his Masters in Simulation Modeling and Analysis from the University of Central Florida in the fall of 2004. His research interests are in scheduling and object-oriented simulation. His email is <awasadikar@mail.ucf.edu>.

KARTHIK NAYARANAN will receive his Masters in Simulation Modeling and Analysis from The Institute of Simulation and Training (IST) in Orlando Florida in the Summer of 2004. His research interests are mainly in networks and simulation. His email is <knayaranan@mail.ucf.edu>.