in Proceedings of ICIS-CET, July 2002, Boston, USA

HIGH LEVEL ARCHITECTURE FOR COLLABORATIVE ROBOTICS SIMULATION ENVIRONMENT

Dr. G. E. Smid, Prof. Ka C. Cheok
Department of Elec. and Syst. Engineering
Oakland University
Rochester, Michigan 48309
Email: smid@oakland.edu

Dr. G. Gerhart, G. Hudas
Robotics Research Group
US-Army TACOM,
Warren, Michigan
Email: grhudas@tacom.army.mil

Abstract

Developing simulation models has become to be a standard practice for developing robotic systems. Simulation techniques provide insight in performance, robustness, dynamic characteristics, and provides a tool to meet design requirements.

A simulation environment is often selected on the basis off short-term needs. For example, the modeling capabilities of the simulator, the availability of the software to the design team, or the familiarity of team members with the tools. However, standardization and the need to exchange simulation models is usually not taken into account.

This paper addresses the need for multiple existing stand-alone simulation models of robotic systems to interact and respond in a synthetic environment. Matlab/ Simulink is used as the target-modeling environment. A toolbox has been developed for Matlab to interface Simulink models with the High Level Architecture (HLA) environment. The following sections discuss the emerging developments in collaborative simulation, and the interfacing aspects and implementation of the toolbox, as well as the design issues from user standpoint.

1 INTRODUCTION

Motivated by the desire to reduce costs and improve quality through interoperability and reuse, the U.S. Department of Defense (DoD) has made significant investments in the area of distributed simulation over the past 15 years. Originating with SIMNET, and evolving through the Distributed Interactive Simulation (DIS) protocols, the Aggregate Level Simulation Protocol (ALSP) and now the High Level Architecture (HLA), DoD has fostered the evolution of standards to support the interoperability of simulations, and the interoperability of simulations and “real world” – e.g. C4I – systems.

The emergence of the World Wide Web (WWW) has produced an environment within which many disciplines are re-evaluating their inherent approaches, techniques and philosophies. The disciplines concerned with computer simulation are no exception to this phenomenon. The concept of ``web-based simulation'' has been introduced and is currently the subject of much interest to both simulation researchers and simulation practitioners within the academic and industrial sectors. Web-based simulation as an area of scholarly pursuit debuted as a 3-paper session at the 1996 Winter Simulation Conference (WSC) and was, by far, the most well attended session within the modeling methodology track of that conference. This success was repeated at WSC ’97. In January 1998 the first conference dedicated to the topic of web-based simulation (WEBSIM ’98) was held as part of the annual Society of Computer Simulation (SCS) Western Multi Conference.

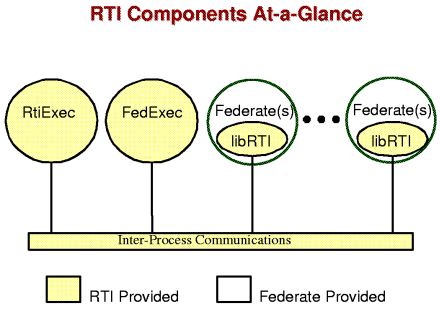


Figure 1 HLA Runtime Infra Structure.

HLA is intended to have wide applicability across the full range of defense simulation applications, including

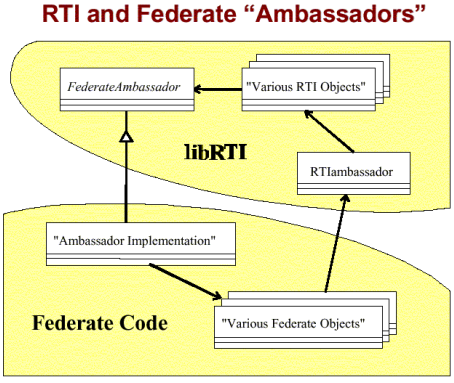


Figure 2 The Ambassador Concept is used to communicate with the HLA RTI.

those used to support training, analysis and acquisition. HLA is designed with a high degree of flexibility, permitting arbitrary mixtures of fidelity and resolution. At the heart of the HLA is the notion of a *federation*.

A federation is a collection of *federates* – simulations and other systems – that interoperate using the protocols described by the architecture. A Federation Object Model (FOM), constructed in accordance with the formalism identified in [1], provides the model specification and establishes a contract between the federates regarding the nature of the activity taking place during federation runtime. Federation execution is accomplished through an HLA Runtime Infrastructure (RTI), which is an implementation of the infrastructure services defined in [3]. In addition to defining services for the RTI, the HLA Interface Specification defines services that must be implemented by federates. Figure 1 depicts the runtime infrastructure and its components in a general fashion for a defined federation.

Modular thinking has introduced a new realm of engineering of a fundamental systems approach. Methods on modularity are diverse and are associated with many disciplines. The common philosophy, is that subsystems can be considered individually, without loss of generality in the overall systems requirements and by providing generalized interoperability and modularity, these subsystems can be re-used in an easy fashion by reconfiguration and reconnection. This is especially true for a simulation scenario that consists of multiple Unmanned Ground Vehicles (UGV's), where each UGV is considered a separate object or module.

It is beyond doubt that simulation performance can be improved, when different modules are executed on different machines. This will allow for multi-rate, multi-processing in an integrated system structure. Furthermore, it will contribute to the generality of the simulator, when attributes to the virtual environment (like terrain, the vehicle platform, optional armor etc.) can be replaced as modules in a multi-purpose simulation facility.

System modularity allows the researcher to extract one module from the simulation, and replace it with another (high detail) module, in order to gain more fidelity and hence more accurate simulation results for the particular subsystem. It also allows for the design and performance evaluation of a particular sub model of a large-scale system, without the need for managing the simulation of the overall system. In other words, to evaluate an obstacle avoidance algorithm for a UGV, the design engineer will have a flexible HLA environment with a UGV and scenery of his/her choice, and be only concerned with the modeling of the obstacle avoidance algorithm. We will discuss the necessary elements of such an environment in more detail.

2 ROBOTICS SIMULATION MODEL

Simulation models essentially provide a prediction of the output values, based on a mathematical formulation of the system characteristics. The level of detail and assumptions, which are associated with the approximation of the real system into the mathematical model, are a critical aspect of the simulation model. They dictate the range of applications for which the model can provide reliable predictions.

In the case of a scenario simulation of multiple UGV's, where it is particularly important to predict the decision behavior of the vehicle, and where the dynamic characteristics are of secondary importance, we could use a forward kinematics model to represent the vehicle.

For example, let us consider the T1-series omni-directional robotic platform, the kinematics model is as follows. Given a speed v and steering φ, we find

$$\begin{aligned} \theta &= \int \varphi \\ X &= \int v\cos(\theta) \\ Y &= \int v\sin(\theta) \end{aligned} \tag{1.1}$$

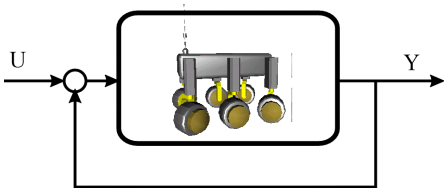
The wheel spin is simply computed by integrating the speed times the radius

$$\omega_n = \int \rho(\varphi)_n R_W v \tag{1.2}$$

where R_W is wheel radius and $\rho(\varphi)_n$ a spin factor relative to the turning radius. According to Ackermann geometry, the steering angle for the wheels is

$$\theta_n = \Delta_n \arctan\left(\frac{\theta}{\cos(\theta)+\delta_n \sin(\theta)}\right). \tag{1.3}$$

where n is the index of the wheel, and δ_n and Δ_n are geometric constants of the vehicle construction. In generalized terms, a UGV model is used to compute new values of the outputs, based on the current outputs Y, internal (electrical, mechanical and software) states, and the inputs U, as



For a regular UGV control, the inputs could represent steering and speed, and the outputs would be the position, orientation and articulation of the vehicle. The steering and speed commands could be generated by a human

operator, or from algorithms for autonomous, semi-autonomous or tele-operated control systems.

For a higher order intelligent UGV, the inputs could represent a target location or a desired trajectory. For these applications, where the UGV contains a higher degree of autonomy, obstacle recognition techniques are commonly used to generate an admissible path towards the target. The information from the obstacle detection sensors and any additional sensors is considered as additional inputs for the UGV model.

3 SIMULATION ENVIRONMENT PROVISIONS

The requirements for a simulation model of a UGV range in a wide variety of applications from detailed dynamic behavior to functional capabilities and training exercises. However, in the unique application of multi-UGV collaborative simulation, the specific objectives for the simulation model focus on reactive behavior and evaluation of the context-based decision structures. For generalization, a minimum necessary set of inputs and outputs for a UGV simulation model has been defined in the previous section. The collaborative simulation environment must provide the capability to handle the data for these inputs and outputs.

Object	Attribute	Type
Obstacle	Location	Double[3]
	Bounding radius	Double[3]
UGV	Position	Double[3]
	Orientation	Double[3]
	Articulation	Double[10][3]
	Steering	Double
	Speed	Double
	Bounding radius	Double
	Mass	Double
	Readiness	Double

Table 1 Object Class and Attribute Definition

The *position*, *orientation*, *steering* and *speed* of the UGV are considered of primary relevance, and the *bounding radius*, *mass*, *readiness* and *articulation* of mobility (i.e. suspension, wheels, steering etc.) of secondary relevance. The ranging sensor input data and the target distance and bearing are considered of primary relevance for a UGV model. Supplemental information on other UGV's like their heading and speed are regarded to be of secondary importance.

To represent the above elementary data in the collaborative simulation environment, two classes of objects have been defined, along with member attributes. Table 1 shows these objects and attributes and the data types.

An individual instance of the UGV object will be created and assigned to each UGV model that is being submitted to participate in the simulation. The High Level Architecture provides for the ownership management of each of the attributes of every instantiated object in the environment. Figure 3 shows a scenario for the ownership management protocol for the HLA environment.

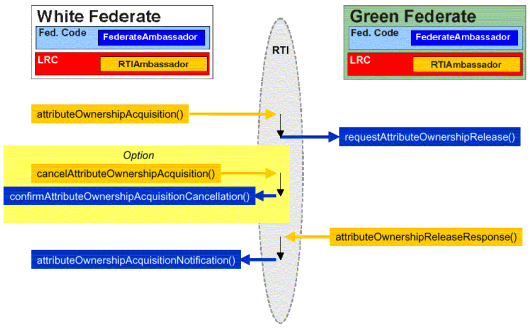


Figure 3 Ownership management in High Level Architecture.

The implementation of the obstacle sensors consists of a query function for the position and range of all obstacle and UGV objects in the simulation environment. The query results can be represented by distance, or by range of bearing. For example: the 10 closest obstacles and their angles; or the obstacles at a relative angle of 0 to 45 degrees of the vehicle and their distance.

4 VIRTUAL REALISTIC ANIMATION

Now that a collaborative simulation environment has been defined, a means of monitoring the simulation results is needed to evaluate run-time performance. For the purpose of full-vehicle simulation, a VRML toolbox has been developed using the Virtual Reality Macro Language protocol, to generate real-time animation of the multi UGV simulation.

Typically the engineer is used to evaluate system's performance using graphs and numerical output. The introduction of powerful 3D environments has made a significant introduction in recent years. Especially for the purpose of vehicle simulations (e.g. simulation of traffic models).

The problem with a multi-user simulation protocol, is that the environment is modified constantly. An animation of the environment from a local workstation would become obsolete soon, when remote simulation participants alternate elements in the environment.

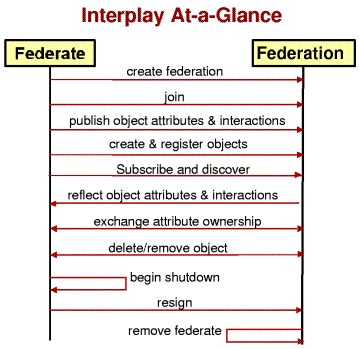


Figure 5 HLA Communication in Typical Scenario.

To overcome this problem, the VRML toolbox has been designed to participate in the environment as an independent simulation model. In this fashion, each user will be able to initiate a 3D animation monitor using the VRML toolbox, and be able to look at a graphical representation of the multi-UGV simulation environment.

5 SIMULATION EXAMPLE I

This example illustrates the evaluation of an path planning and obstacle avoidance algorithm for an autonomous UGV. The application of simulating decision algorithms required the mathematical modeling of several components of the system. Not the least important are the models for the terrain environment, the UGV kinematics or dynamics and the vehicle-terrain interaction.

Therefore, first a kinematics model of a UGV is being simulated in workstation A. During initialization, the simulation environment creates an instance of the object UGV in the UGV simulation federation, with for example the name 'T1'. The simulation then becomes the owner of the attributes that are listed in Table 1.

A second workstation B executes an obstacle and terrain model, by maintaining location and radius of a series of objects in the simulation environment. During initialization, each obstacle will be assigned a name and attributes which are owned by station B.

The third workstation C executes the path planning and obstacle avoidance algorithms. During initialization this simulation execution will look for 'T1' in the UGV federation, and will aim to acquire the ownership of its attributes for *steering* and *speed*.

By applying the obstacle avoidance and target tracking procedure on the obstacle data from workstation B and the UGV system data from workstation A, this simulation model will generate appropriate steering and speed commands for the UGV and attach these commands to the UGV model.

6 SIMULATION EXAMPLE II

The usefulness of the HLA architecture becomes apparent when designing the control model for the tracking algorithm where multiple vehicles are involved. Separate workstations will be used to execute the vehicle kinematics models, the operator interface and the scenery environment with obstacles and roads. The design workstation will contain only the tracking algorithm, and use the HLA toolbox modules to interface the algorithm with the follower vehicle.

This example considers a scenario of two vehicles in the application of Leader-Following [12]. The first vehicle (i.e. Lead vehicle) is controlled by a human operator. The simulation model captures the operators steering and speed commands and applies them to the dynamics model of the lead vehicle.

The second vehicle (the Follower vehicle) is being controlled by a tracking / obstacle avoidance algorithm. This simulation model is executed on a separate workstation. During initialization it creates a second instance of a vehicle, now with the characteristics and attributes of the follower UGV. The steering and speed commands are generated from the tracking algorithm.

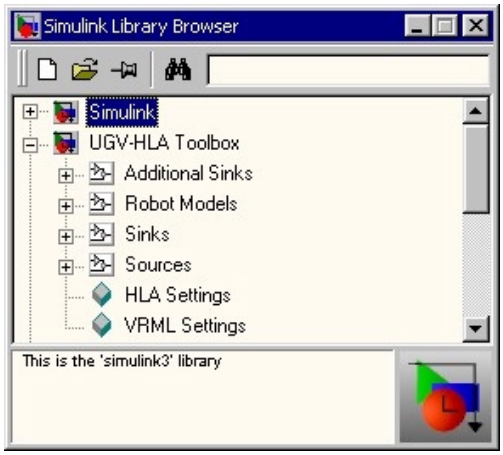


Figure 4 UGV - HLA Library in Simulink

7 IMPLEMENTATION IN SIMULINK

The implementation in Simulink requires an interface to exchange information and data with the High Level Architecture collaborative environment, during the simulation process. For this purpose, a dedicated Simulink Toolbox has been developed, which contains blocks that can be dragged into any model, providing access to UGV models, obstacles and terrain characteristics. Table 2 shows the simulation procedure calls in the Simulink library interface. During each of these function calls, several tasks must be executed to collaborate with the HLA environment. The custom configurations for a model to interface as a federate to the

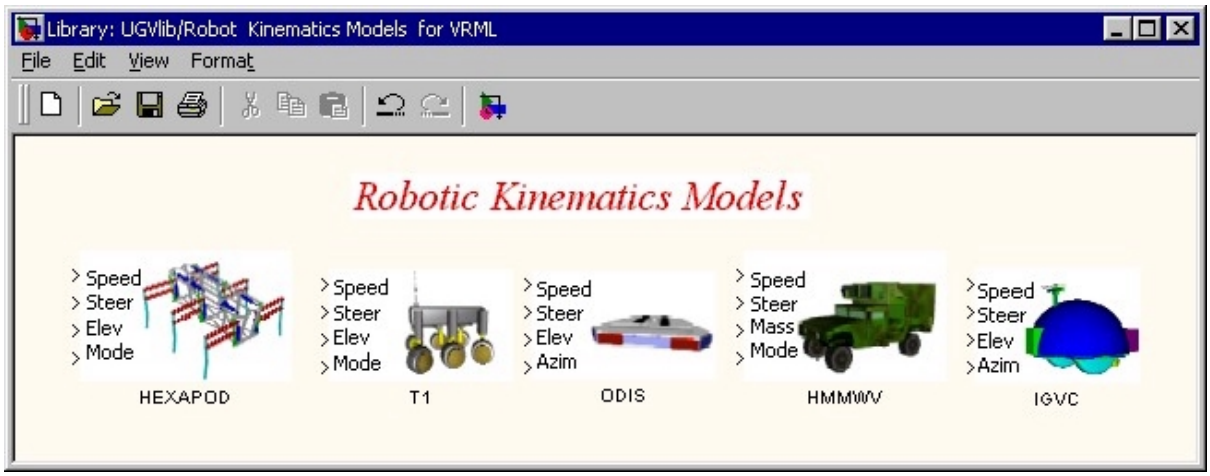


Figure 6 Robotics Models Library

Simulink Function Prototype	**HLA Protocol Task**
INITIALIZE SIMULINK	1. Check each block for simulation settings

START SIMULATION
`mdlInitializeSizes()`
`mdlInitializeSampleTimes()`
`mdlInitializeConditions()`
	1. If federation does not exist, create it
	2. Join federation execution as new federate
	3. Initialize time management
	4. Publish and subscribe simulation object types
	5. Check if objects (node name) exist.
	If a name already exists, then for:
	• Output node, aquire ownership
	• Through node, conditionally aquire
	• Input node, subscribe to data updates

SIMULATION LOOP `mdlOutputs()`	1. Process ownership transactions (handle requests) 2. If object is owned, update new data values 3. Process updates for objects 4. Return object data to Simulink 5. Advance federate simulation time with federation
STOP SIMULATION `mdlTerminate()`	1. If object is owned and was created, then destroy 2. If object is owned and aquired, then release ownership 3. Process ownership transactions 4. Unpublish and unsubscribe object types 5. Resign from federation and destroy federate
TERMINATE MATLAB (OR CLEAR FCN) `CleanUp()`	[1] Attempt to destroy federation execution. (The last federate to resign from the federation execution will succeed in this attempt)

Table 2 HLA Task specification for Simulink interface model.

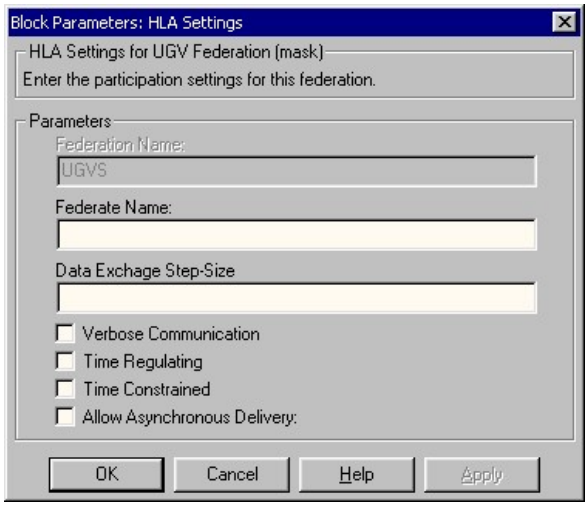


Figure 7 The HLA Configuration Interface in Simulink.

HLA federation have been made accessible from the Simulink Interface through a special block "*HLA Settings*". The features of this configuration interface are displayed in Figure 7.

8 CONCLUSIONS

The HLA collaborative protocol provides powerful capabilities for conducting multi-UGV simulation scenarios.

Unfortunately, the protocol lacks a convention for referencing the validity and fidelity of simulation models. As mentioned in the introduction, each simulation model is subject to approximations and assumptions. Without prior knowledge of these simulation model properties, no conclusions can be drawn from the results of the experiments.The implementation of the HLA interface in Matlab/Simulink provides a convenient and user-friendly design environment to conduct experiments.

9 ACKNOWLEDGEMENTS

The authors like to acknowledge the Research Business Group at US-Army Tank Automotive and Armaments Command (TACOM) for the support and collaboration. In particular Dr. G. Gerhart, Mr. G. Hudas and also Dr. J.L. Overholt and Dr. D. Gorsich for the illuminating discussions. The work described in this paper has been conducted under DoD contract DAAE07-01-Q-BAA1.

10 REFERENCES

[2] High Level Architecture Object Model Template. Technical Report, Version 1.3. U.S. Department of Defense, 1998.

[3] High Level Architecture . Technical Report, Version 1.3. Department of Defense, 1998.

[4] High Level Architecture Interface Specification. Technical Report, Version 1.3. U.S. Department of Defense, 1998.

[5] The Modeling Methodological Impacts of Web-Based Simulation. Proceedings of the 1998 SCS International Conference on Web-Based Modeling and Simulation, San Diego. E. H. Page and A. Buss and P. A. Fishwick and K. J. Healy and R. E. Nance and R. J. Paul, 1009. pp.123-128.

[6] Virtual Vehicle System Simulation. A Modular Multi-CPU Approach in Real-Time. Ph.D. Thesis, G. E. Smid, 1999, Oakland University.

[7] Multi-Conputer Real-Time Simulation of Vehicle Control Systems. G. E. Smid and Ka C. Cheok. Proceedings of International Conference on Intelligent Systems (ICIS) 1998 Paris, France.

[8] Distributed Simulation with {JavaGPSS} Based on the High Level Architecture. Klein and U. S. Strassburger and J. Beikrich. Proceedings of the 1998 {SCS} International Conference on Web-Based Modeling and Simulation, pp. 85-90. San Diego, CA 1998

[9] The Rise of Web-Based Simulation: Implications for the High Level Architecture. E. H. Page. Proceedings of the Winter Simulation Conference 1998, pp. 1663-1668

[10] Zero Lookahead and Repeatability in the High Level Architecture. R. M. Fujimoto. Proceedings of the 1997 Spring Simulation Interoperability Workshop, Orlando, FL 1997.

[11] The Department of Defense High Level Architecture. J. S. Dahmann and R. M. Fujimoto and R. M. Weatherly. Proceedings of the 1997 Winter Simulation Conference 1997, New York, NY. Pp. 142-149

[12] A Fuzzy Logic Intelligent Control System Architecture for an Autonomous Leader-Follower Vehicle. Ka C. Cheok and G. E. Smid and K. Kobayashi and J. L. Overholt and P. Lescoe. Proceedings of American Control Conference (ACC) 1997, Albuequerque, New Mexico.

[13] A Fuzzy Logic Hierarchical Intelligent Control System Paradigm for an In-Line-Of-Sight Leader-Follower HMMWV. Ka C. Cheok and K. Kobayashi and G. E. Smid and P. Lescoe and J. L. Overholt. Journal of Robotic Systems 1997, Vol. 14, No.6, pp.407-420

[14] A Simple Vehicle Dynamics Modeling by the Object Oriented Approach. K. Kobayashi and K. Watanabe and Ka C. Cheok and G. E. Smid. SICA, 1997.

[15] In-Line-Of-Sight Leader-Follower HMMWV's with Fuzzy Logic Preview Control. Ka C. Cheok and K. Kobayashi and G. E. Smid and P. Lescoe and J. L. Overholt. Proceedings of AUVSI, 1996, Orlando, FL.

[16] Providing Conceptual Framework Support for Distributed Web-Based Simulation Within the High Level Architecture. E. H. Page and S. P. Griffin and S. L. Rother. Proceedings of {SPIE}: Enabling Technologies for Simulation Science {II}, 1988.