Rapid Synthesis of Multi-Model Simulations for Computational Experiments in C2

H. Neema, H. Nine, G. Hemingway, J. Sztipanovits, G. Karsai
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN, USA
Email: { himanshu, hnine, heminggs, sztipaj, gabor } @isis.vanderbilt.edu

Abstract-Virtual evaluation of complex command and control concepts demands the use of heterogeneous simulation environments. Development challenges include how to integrate multiple simulation platforms with varying semantics and how to integrate simulation models and the complex interactions between them. While existing simulation frameworks may provide many of the required services needed to coordinate among multiple simulation platforms, they lack an overarching integration approach that connects and relates the semantics of heterogeneous domain models and their interactions. This paper outlines some of the challenges encountered in developing a command and control simulation environment and discusses our use of the GME meta-modeling tool-suite to create a model-based integration approach that allows for rapid synthesis of complex HLA-based simulation environments.

I. INTRODUCTION

Evaluation of emerging command and control (C2) concepts necessitates a sophisticated modeling and simulation infrastructure that allows for the concurrent modeling, simulation and evaluation of (1) the C2 system architecture (advanced system-of-systems modeling), (2) the battle space environment (scenario modeling and generation), and the (3) human organizations and (group and individual) decision making processes (human performance and man-machine interaction modeling). The complexity of this issue demands the use of a multitude of existing simulation tools interacting in a coordinated environment.

Issues encountered in developing this environment relate to the integration of various simulation platforms, each with its own semantics, and integration of the models that execute on the platforms and the interactions between models. Integration frameworks, such as the High-Level Architecture (HLA) ([1], [11], and [25]) provide APIs that have helped to reduce the complexity of developing simulations that span multiple different platforms, but many challenges remain in developing and deploying these simulations. Pervasive use of models and integration at the domain-specific model level can decrease the effort required to integrate multiple platforms and increase the adaptability of the resulting environment, though this approach is still not without its own set of challenges.

Each simulation platform, when incorporated into the overall simulation environment, requires integration at both the API level and at the semantic or interaction level. API integration provides basic services such as time coordination, message passing, and shared object management. Semantic integration is more subtle and depends upon the context of the overall simulation environment.

The explicit use of models throughout the simulation environment opens the door for model-based integration techniques. The C2 Wind Tunnel environment uses the Generic Modeling Environment (GME) [2] meta-modeling tool chain to integrate the operational semantics of multiple simulation platforms, to manage the configuration and deployment of scenarios into the simulation environment, and to generate necessary integration code for each integrated platform.

The GME tools use a generic modeling environment to facilitate graphical development of domain-specific modeling languages (DSML). These languages can express both the structural and operational semantics of a model. The C2 Wind Tunnel project has developed DSMLs that capture the semantics of each integrated simulation platform and an overarching DSML to model the interactions between simulation models. We have written custom interpreters to dynamically import and export platform-native models into the GME DSMLs and to generate HLA configuration files and glue code. These tools allow our environment to be rapidly reconfigured or extended, furthering the goal of rapid evaluation of emerging command and control concepts.

The organization of the remainder of this paper is as follows. The next section discusses the work related to our research. Section III details the C2 Wind Tunnel simulation framework and our approach for model integration using the GME environment. Section IV reviews the details of the integration of several simulation tools. Section V

covers results from using the framework in a real-world scenario. The final section concludes the paper and outlines the planned future work.

II. RELATED WORK

There has been some work to integrate a set of simulation packages with or without HLA. For example, one of the previous efforts at our institute, that relates to heterogeneous simulation, particularly of embedded systems, is the MILAN framework [10]. While efforts that relate to integrating simulation packages with HLA include OPNET [12], MATLAB-HLA integration methods [13], SLX integration [14], JavaGPSS integration [15], DEVSJAVA [16], [17], and PIOVRA [18]. Also, there is some amount of commercial integration software available, including HLA Toolbox [19] for MATLAB federates by ForwardSim Inc. [20] and MATLAB and Simulink interfaces to HLA and DIS based distributed simulation environments [21] by MÄK Technologies [22]. Additionally, there also have been some efforts on enhancing HLA support by complementary simulation methodologies such as in [23], and [24]. However, most of these efforts pursue HLA integration of isolated simulation tools, do not exploit HLA for distributed simulation, or are commercial applications for a dedicated simulation tool. Moreover, these efforts (except MILAN) do not have any support for model-based rapid integration of simulation tools.

In contrast to the current research efforts, we have focused on developing a model-based integration framework to integrate a range of simulation tools such as MATLAB/Simulink, OMNeT++, DEVSJAVA, CPN Tools, Delta3D, and legacy Java, C, or C++ HLA-federates, and to rapidly synthesize large-scale HLA-based heterogeneous simulation. To that end, this paper gives a detailed description of our approach demonstrating rapid model-based integration of various simulation tools.

III. MODEL-BASED SIMULATION ENVIRONMENT

Complex command and control simulations require coordination between multiple existing simulation platforms. The HLA provides a standard for a Run-Time Infrastructure (RTI) that supports the coordinated execution of distributed simulations. However, integration of the different platform-specific simulation models remains a challenging problem. Our project introduces a new model-integrated approach for the simulation integration problem. The primary contribution of this effort is the development of an overarching modeling environment, based on the GME, and a suite of model transformations to synchronize between multiple platform-specific models.

The composition, deployment and execution of the entire simulation is configured using the central modeling environment. A large set of well-known simulation tools are supported for heterogeneous simulations. We provide reusable run-time components to coordinate the execution of the various domain-specific models. These components are configured from the modeling environment with a set of XML files. In some cases, both glue code and configuration files are generated for configuring the simulation tool.

The *virtual* component models in the C2 Wind Tunnel are integrated using the GME environment. Model transformations, executed using custom interpreters, generate the necessary configuration information and glue code. These pieces are incorporated into each of the different simulator platforms. At run-time, the platforms interact using the HLA infrastructure.

In the following sub-sections, we briefly describe the HLA and the GME tool-chain, and then we discuss in detail the integration approach and life-cycle of the entire environment.

A. *HLA AND GME BACKGROUND*

The High-Level Architecture ([1], [11], and [25]) is a standard framework for distributed computer simulation systems. It was originally developed by the U.S. Department of Defense and is their required standard for simulation interoperability. IEEE now oversees the ongoing evolution of the architecture as HLA Standard 1516. Communication between different federates is managed by the Run-Time Infrastructure (RTI). The RTI provides a set of services such as time management, data distribution, and ownership management. Other components of the HLA standard are the Object Model Template (OMT) and the Federate Interface Specification (FIS).

Since HLA is an accepted standard, a number of commercial, academic, and alternate RTI implementations are available. Currently, we use the Portico RTI [4] – which provides support for both C++ and Java clients and is compliant with version 1.3 of the HLA standard.

Command and control simulations are composed of a variety of simulation tools such as Matlab/Simulink, CPN Tools, and OMNeT++. Each of these simulation tools has its own simulation clock and methods for progressing the simulation forward through time. The HLA interface specifications - HLA-interactions and HLA-objects (shared objects) - form the basis of communication among these

heterogeneous simulation tools. An interface specification for a federation consists of a definition for each of the interactions and objects that are to be supported by the RTI. This is typically specified in a textual language that is specific to the chosen RTI. A graphical representation of the same information is more comprehensible and readable and it allows for capturing a variety of configuration elements that are necessary for the synthesis of heterogeneous simulations. The C2 Wind Tunnel captures not only the necessary interface specification for running heterogeneous simulations over HLA, but also a variety of mechanisms for configuring, enhancing, and detailing the simulation execution.

The Generic Modeling Environment is a meta-programmable model-integrated computing (MIC) toolkit that supports creation of rich domain-specific modeling and program synthesis environments. Configuration is accomplished through metamodels specifying the modeling paradigm (modeling language) of the application domain. A metamodel is used to define all the syntactic, semantic, and presentation information regarding the domain.

B. C2 WIND TUNNEL META MODEL

Below, we describe briefly the C2 Wind Tunnel metamodel for HLA integration of federates running in the C2 Wind Tunnel framework. This is used to automatically configure GME as a modeling environment that supports modeling heterogeneous simulations to run on the C2W framework.

Fig. 1 shows the primary portion of the metamodel that defines the universe of composition elements of an HLA federation running in the C2 Wind Tunnel framework and the set of communication elements for inter-federate communication.

As shown in Fig. 1, the three main elements in a federation (defined by the model *FOMSheet*) are *Interaction*, *Object*, and *Federate* representing an *HLA-Interaction*, *HLA-Object*, and an *HLA-Federate* respectively. Note that the proxy elements are simply references to their respective main models. Federates in an HLA federation communicate among each other using HLA-interactions and HLA-objects – which are in turn managed by the RTI. Interactions and objects, in an analogy with inter-process communication, correspond to message passing and shared memory respectively. As seen in Fig. 1, the metamodel fully supports the key attributes of these communication elements such as delivery method, message order (timestamp or receive order), and parameters. The main attribute of a federate, as far as HLA-based synchronization is concerned, is its *Look-ahead* – the period of time in the future during which the federate guarantees that it will not send an interaction or

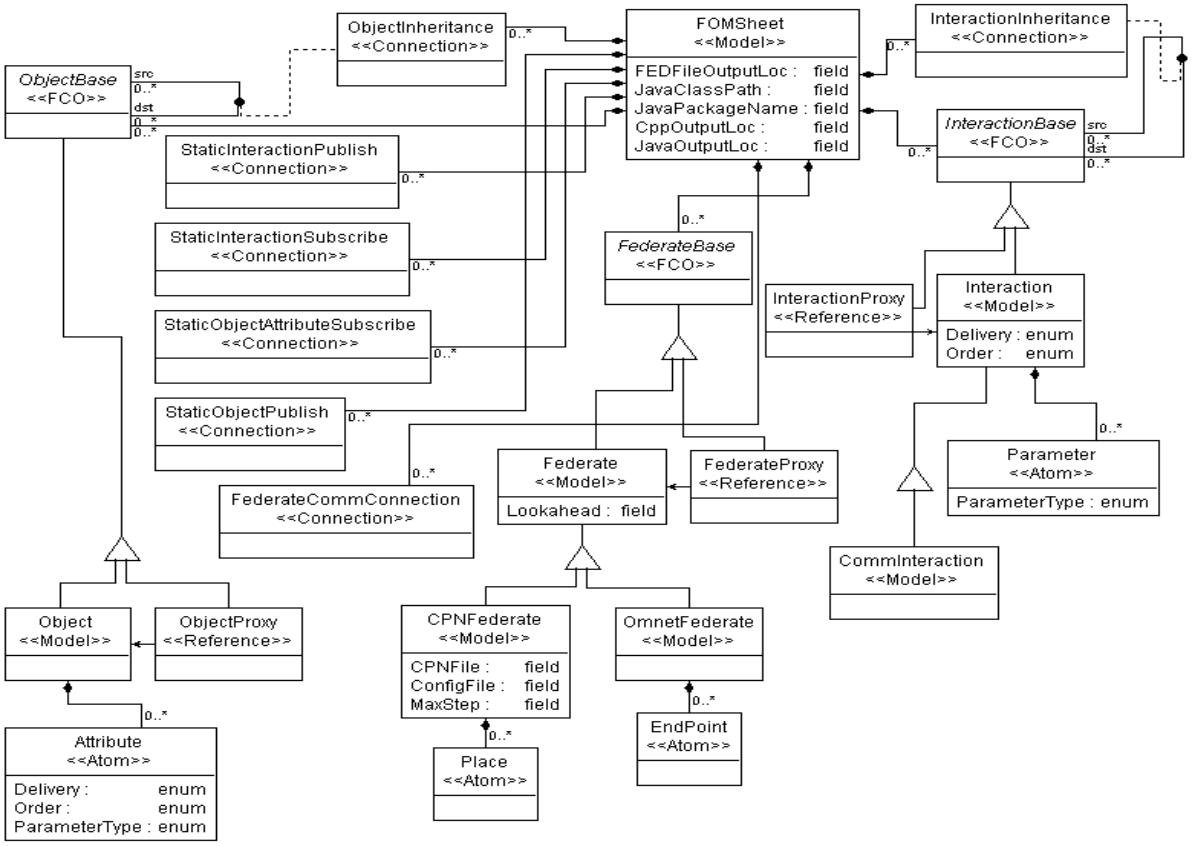

Fig. 1: Part of the C2 Wind Tunnel Meta Model

update an object. Further, we define specialized federate elements for simulation tools requiring special integration models, such as CPN Tools and OMNeT++, and define special communication elements required by them, viz. *Place* and *EndPoint* respectively. We discuss model-based integration of these special simulation tools in subsequent sections. The metamodel goes an additional step forward by also providing full support for defining the means to publish and subscribe elements required for the heterogeneous simulation. This is accomplished by defining special publish and subscribe connection elements for both interactions and objects. Lastly, the attributes of FOMSheet captures the names and location for configuration code that enables the integration of a range of supported simulation tools. We will be describing this capability in detail in a following section. Model-based integration of OMNeT++, Delta3D, Matlab/Simulink, CPN Tools, and DEVSJAVA is discussed in Section 4.

C. *Integrating Simulation Models*

Two main aspects, the data representation and the data flow, are common elements of most of the domain-specific simulation modeling paradigms – so these are the key points of our integration models. The integration models describe both the data representation and data flow elements and, in some cases, include special elements as the placeholders for domain-specific models.

The data representation models consist of interaction and object models. Interactions are stateless and can have parameters while objects have states – which are represented as a set of attributes. Both interactions and objects can have an inheritance hierarchy. These data representation models directly map to the HLA Federation Object Model (FOM). Fig. 2 shows an example of the data representation models with an interaction class-inheritance tree on the top, and an object class-inheritance tree on the bottom.

Once the data representation models are created the modeler can define publish-subscribe data flow relations by creating federates and connecting them to interactions or object attributes with directional links. Federates can publish or subscribe interactions or object attributes.

To aid in organization, the integration model can be broken down into sheets. This type of partitioning is up to the modeler and does not directly affect the semantics of the model. The GME modeling language supports references and proxies to connect the sheets. In this manner, large models can be visualized and edited easily. In addition, libraries of such models can be built and reused later.

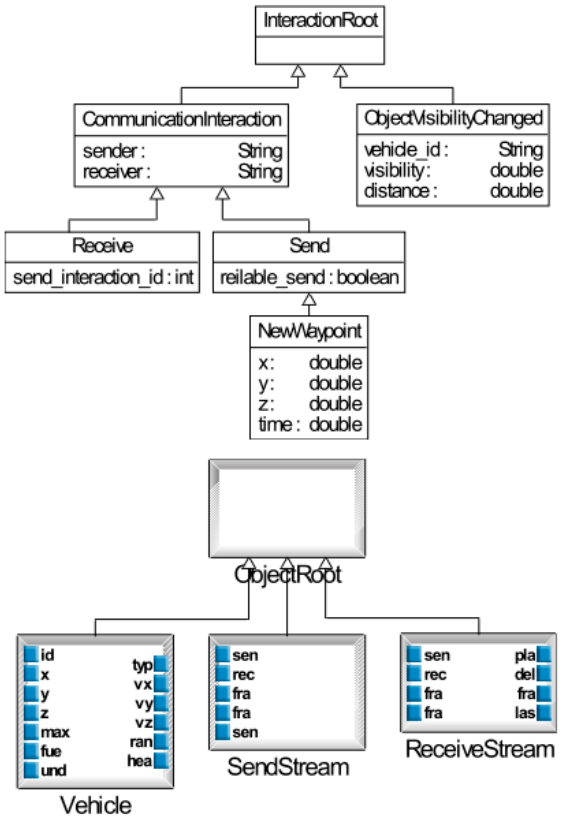

Fig. 2: Data Representation Model Examples

D. *The Development Cycle*

The GME supports a plug-in architecture to add model operator modules called interpreters. These interpreters can access the model through a high-level easy-to-use interface, the Builder Object Model (BON). Interpreters can be written in either C++ or Java. The C2 Wind Tunnel comes with a set of interpreters that support the transformation between domain-specific and integration models, and the generation of run-time configuration information.

In many cases the domain specific models already exists, so data representation and data flow models can be easily imported. In other cases, we need to export these from the integration models. An iteration of these steps can be performed until the models are synchronized. An example of this is presented in the Colored Petri Nets section.

Several integration-related parameters can be set in the modeling environment. One example is the static look-ahead value of a federate. This value can be set manually if it is not possible to calculate it automatically.

After finalizing the models and setting the parameters, an interpreter generates all the necessary configuration information. This includes the HLA .fed file, XML and other configuration files for the domain specific federates. Where necessary, Java or

C++ skeleton code, based on the integration models, is generated for run-time use in federates.

At times, the modeling environment itself may need some extensions when either a change in metamodel is made or integration glue code needs to be updated. An example of this type of change would include addition of support for a previously unsupported simulation platform. For this, slight extensions to the metamodel (though rarely) may need to be made (e.g., to define new interactions or objects for the new simulation platform). In addition, the interpreter needs slight extensions for generating glue code for the new simulation platform. The current infrastructure enables one to accomplish support for newer simulation platforms with much ease and speed.

IV. SUPPORTED DOMAIN SPECIFIC MODELS

This section describes the domain specific simulation platforms currently supported by the C2 Wind Tunnel. These platforms are OMNeT++, Delta3D, Matlab/Simulink, and Colored Petri Nets. This set was selected to support a wide range of unmanned aerial vehicle command and control scenarios.

A. OMNeT++ - COMMUNICATIONS FEDERATE

In a command and control simulation environment it is essential to model and simulate the communication network in order to study mission critical situations like network failures or network attacks. After evaluating multiple public domain network simulators against project requirements, OMNeT++ [6] was selected as the network simulation platform.

OMNeT++ provides high performance, faithful protocol simulation above the physical layer. Its modular architecture allows for easy replacement of the event scheduler in order to integrate with HLA, obviously a key requirement. OMNeT++ does not use computationally expensive radio-signal propagation models; instead, it simulates the physical layer with simple probabilistic models.

Even in the case of a small network, simulating the full protocol stack generates a large number of messages, which would flood the simulation and decrease performance if they were dispatched to the RTI. This approach does not scale well for large numbers of communicating actors. Finding the optimal tradeoff between scalability, speed and simulation accuracy is an important issue.

We developed a tool, *NetworkSim*, which is an HLA-compliant reusable communication network simulator based on OMNeT++. NetworkSim provides a set of high-level communication protocols (reliable send, streaming) while internally it faithfully simulates the full network stack. This approach allows for application level interactions of interest to be captured while minimizing HLA traffic.

The NetworkSim scheduler keeps the RTI and OMNeT++ clocks synchronized. The advanceTime() call inside receives interactions from the RTI and a dispatcher mechanism converts them to the appropriate protocol module which interprets it and can schedule new OMNeT++ messages.

B. DELTA3D - VISUALIZATION, SENSOR DATA, AND PHYSICS FEDERATE

Delta3D [3] is an actively developed open-source project that integrates numerous modules for visualization, communications, and dynamics. It provides native HLA integration and scene modeling tools. Because of its native HLA integration, no GME interpreter is needed to integrate it within the C2 environment.

Delta3D-based federates play several different roles in the C2 Wind Tunnel. The first role is to provide 3D visualization of the virtual world as the simulation executes. The second role is to simulate the sensor data being collected by cameras mounted on each UAV. The real-time video data is sent back to an operator station. The logical packets are sent through the simulated network via HLA interactions and objects shared between Delta3D and OMNeT++, but the actual image data is not transferred. It is regenerated on the operator side since the network bandwidth is more of a bottleneck in this situation than CPU load. Using this approach, the transmitted video stream can be faithfully simulated and the solution remains scalable as well. The final use of Delta3D federate is as the physics simulator handling tasks such as collision detection.

C. MATLAB/SIMULINK - UAV PLANT AND CONTROLLER FEDERATE

Simulink [5] is a widely used simulation environment for modeling dynamic and embedded systems such as communications, controls, and signal processing. It uses a set of pre-built block libraries for designing and controlling the simulation.

In our scenarios, we use X4 Simulink models for simulating the dynamic behavior of Unmanned Aerial Vehicles (UAVs) with four rotors. The simulation model is a high-fidelity dynamics model with a range of parameters to setup the simulation such as initial location, speed and acceleration vectors, yaw, pitch, and roll. Additionally, the model takes as input a set of time-stamped waypoints that allow the model to calculate the trajectory of the UAV to reach the waypoint in the given amount of

time. The model can then calculate the flight parameters including speed and turns to find a shortest-path that makes the UAV reach the waypoint at the required time. At run-time, the Simulink model provides continuous outputs of the current position of the UAV as well as its current flight parameters such as yaw, pitch, and roll.

Our UAV federate subscribes to *NewWayPoint* interactions and publishes *PosUpdate* interactions. Once modeled in GME, the interpreter for the model generates code that can be used to directly link the Simulink model with the C2 Wind Tunnel framework.

In addition, we have written four generic Java classes that are used for integration of any Simulink model. These are *Matlab-Federate*, *Matlab-HLA-Bridge-Base*, *Matlab-Interaction*, and *Matlab-Interaction-Parameter*. The class *Matlab-Interaction-Parameter* is simply a container class to hold a variety of supported data types as RTI interaction parameters. The class *Matlab-Interaction* captures interactions sent to or from the RTI and contains a non-null parameter representing the ID of the interaction and an array of supplied *Matlab-Interaction-Parameters*. The abstract class *Matlab-HLA-Bridge-Base* provides interfaces for converting between *Matlab-Interaction* and *HLA-Interaction*. Finally, the abstract class *Matlab-Federate* encapsulates interfacing with the HLA for initializing the federate, its Look-ahead, interactions models used, and it's publish and subscribe relationships with other federates in the federation. Additionally, the Matlab-Federate class serves as a mediator for communication with other federates via the HLA by providing a number of methods that are accessible from MATLAB.

A custom GME interpreter generates a set of Java and MATLAB files. The generated class *Matlab-<FedName>* is derived from the generic class *Matlab-Federate*. It implements the method for registering the published and subscribed interactions and provides wrapper code for send and receive interaction methods and simulation control methods. The generated class *Matlab-<FedName>-Matlab-HLA-Bridge* is derived from the generic class *Matlab-HLA-Bridge-Base*. It populates the used interaction types and implements the base class methods for converting interaction parameters between HLA and C2 Wind Tunnel framework types. Additionally, the interpreter generates a pair of S-functions for the input and output bindings from Simulink (viz., *Matlab-<FedName>-Receiver.m* and *Matlab-<FedName>-Sender.m*) respectively. To control the Simulink clock and to provide input and output bindings, these S-functions use block methods *mdlGetTimeOfNextVarHit*, mdlUpdate, and mdlOutput.

D. CPN - ORGANIZATIONAL DECISION MAKING FEDERATE

C2 Wind Tunnel scenarios use Colored Petri Nets (CPN) to model and simulate human decision-making organizations. We use the CPNTools [8] environment augmented with the BRITNeY [9] extension tools. This extension provides access to lower level functionality, which was necessary for integration.

The primary challenge while integrating CPN into the C2 environment was correct time synchronization. To ensure the CPN model execution stops at desired times an extra place and a transition, which is set to fire with a predefined frequency, was added. The CPN model optimistically progresses ahead of the HLA clock, but when needed, it can be rolled back to a desired time.

CPN models can be imported into integration models with a GME model interpreter. Upon importing a CPN model, a CPNFederate model is created in the integration model and the CPN places show up as ports of the CPNFederate. The ports can be connected to the other parts of the integration model to specify inputs and outputs for the CPN. Fig. 3 shows the integration steps.

In the configuration phase, an XML file is created which describes the input-output bindings. The run-time CPN execution engine will read this file and simulate the CPN according to the specification. The set of places to monitor during execution can also be specified.

V. EXPERIMENTATION WITH THE PLATFORM

The C2 Wind Tunnel project's goal is to create a heterogeneous simulation environment tailored towards scenarios involving the interaction of multiple unmanned aerial vehicles, their operating conditions, and the associated command and control organization and infrastructure. Rapid evolution of scenario details and easy evaluation of command and control effectiveness are key motivators. A typical scenario involves the deployment of one or more UAVs (implemented using Simulink X4 models) into a combat zone. The deployment zone and the physical dynamics of the ground and aerial vehicles are modeled and visualized using Delta3D. The UAVs may have objectives including data collection, target acquisition and engagement, or battle damage assessment. Sensors (implemented using a Java federate) mounted on the UAVs must collect information and relay it via a communications network back to a centralized decision making

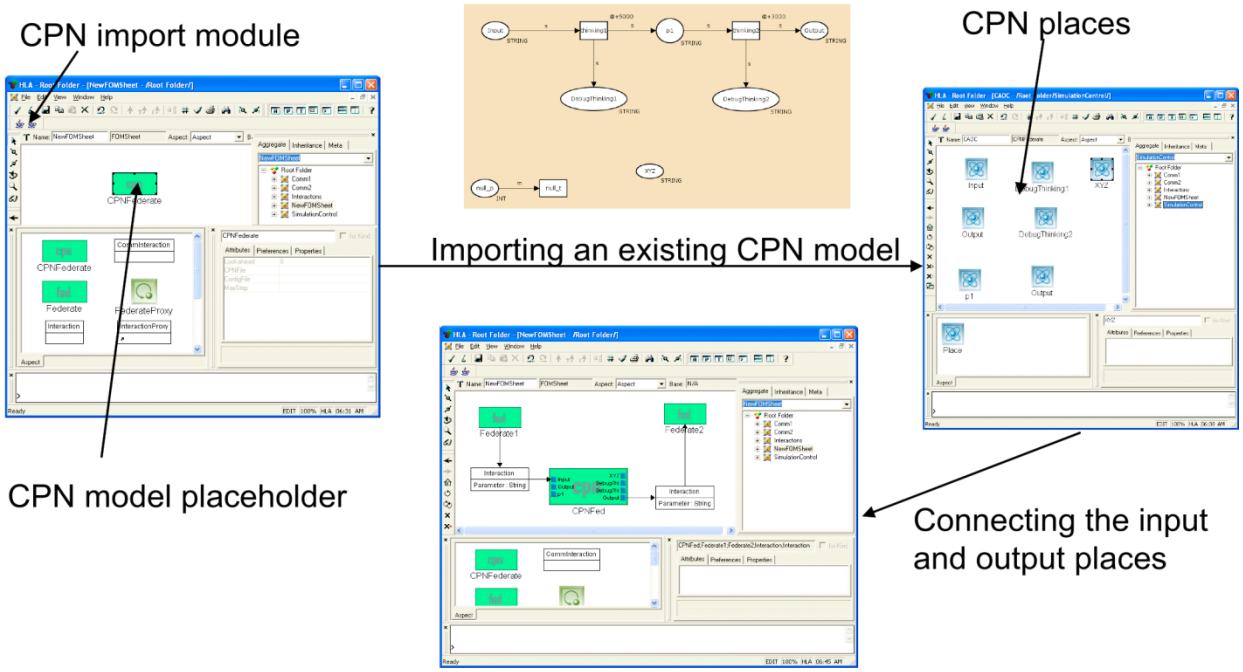

Fig. 3: CPN Model Integration Steps

organization (a CPN model). The organization must react appropriately to the information and provide guidance to the vehicles. In addition, the UAVs are themselves highly autonomous and must utilize collected sensor data to pursue their given objectives. Out first goal was to demonstrate that we could rapidly synthesize such simulations using our model-based integration environment involving all the domain-specific simulation platforms described above. For demonstration purposes, we used an urban scenario with four blue UAVs and two red ground vehicles. All UAVs have video sensors and are continuously transmitting video data to the control station. The control station remotely controls the UAVs in a formation flight and assesses the targets one by one based on the initial position estimates of the targets.

All network communications are simulated using the OMNeT++ federate. We tested how a network attack affects mission performance. The abstraction of low-level physical layer communication in OMNeT++ makes it straightforward to implement network attacks. Attacks that we found most reasonable in our context are Distributed Denial of Service (DDOS) attacks.

Using a collection of "zombie" nodes in dedicated sub-networks, either parameterized or controlled by a master node, any type of "dumb" or non-adaptive attacks can be simulated. For our purposes, DDOS was sufficiently disruptive. Each zombie machine sends, at specified intervals, service requests to every discovered valid node. As communications degrade, the effect on formation flight is pronounced. The formation flight does not loosen; rather it collapses altogether. The UAVs continue in their individual directions entirely. Sensor information is lost, so the operator becomes unable to repair the loss manually.

The impact of these attacks on the command and control of the UAVs closely mirrors both the theoretical and real-world consequences observed previously in other contexts. This gives the experimenters confidence that the results of simulation can be directly applied to the modeled scenarios.

VI. CONCLUSION AND FUTURE WORK

Once all of the C2 Wind Tunnel federates were integrated via the HLA and into the GME environment, the team could begin exploring possible scenarios. A significant motivator for the overall project is to be able to rapidly explore "what-if" scenarios and evaluate both the organizational decision-making and other impacts such as network communications disruptions.

The use of GME to integrate domain-specific models from each simulation platform and generate all of the needed configuration and integration code dynamically greatly reduces the time required to modify a base scenario. Changing the number of UAVs, the paths of virtual environment adversaries,

or the time, severity, and duration or network attacks require simple parameter modifications. Even a range of federates running on different simulation platforms can be instantiated dynamically. GME handles the regeneration of all the remaining code and configuration files for each of the affected federates.

In the future, we expect to integrate additional simulation platforms to expand the range of possible scenarios. Consequently, new simulation platforms will require expanding the existing integration metamodel. These enhancements should allow for greater scenario flexibility and reduced development, configuration, and operational costs. Additionally, we are exploring alternatives to implement capabilities in the C2W framework to alter and configure the entire simulation at run-time.

VII. ACKNOWLEDGMENTS

The authors acknowledge financial support from the US Air Force Office of Scientific Research (Grant No. FA9550-06-1-0267) under the project "Human Centric Design Environments for Command and Control Systems: The C2 Wind Tunnel."

The work described in this paper was inspired and significantly influenced by our collaborators, Prof. Alexander H. Levis, Prof. Lee W. Wagenhals, and Prof. Abbas K. Zaidi from George Mason University. Authors acknowledge contribution and help from Mr. Mike Kretzer, AFIOC and Mr Timothy Busch, AFRL-RI.

VIII. REFERENCES

[1] HLA standard – IEEE standard for modeling and simulation (M&S) high-level architecture (HLA) – framework and rules. ieeexplore.ieee.org/servlet/opac?punumber=7179.

[2] Sztipanovits, J., and Karsai, G. 1997. "Model-Integrated Computing", *IEEE Computer*,V.30. pp. 110-112.

[3] Delta3D - www.delta3d.org.

[4] Portico RTI - www.porticoproject.org.

[5] Matlab/Simulink - www.mathworks.com.

[6] OMNeT++ - www.omnetpp.org.

[7] DEVSJAVA - www.acims.arizona.edu.

[8] CPN Tools - wiki.daimi.au.dk/cpntools/cpntools.wiki.

[9] M. Westergaard: "The BRITNeY Suite: A Platform for Experiments" in Proceedings of Seventh Workshop on Practical Use of Coloured Petri Nets and the CPN Tools. Århus, Denmark, October 2006.

[10] Agrawal A., Bakshi A., Davis J., Eames B., Ledeczi A., Mohanty S., Mathur V., Neema S., Nordstrom G., Prasanna V., Raghavendra, C., Singh M.: MILAN: A Model Based Integrated Simulation.

[11] Dahmann, J.S., R.M. Fujimoto, and R.M. Weatherly. 1997. The Department of Defense High Level Architecture. In Proceedings of the 1997 Winter Simulation Conference, ed. S. Andradóttir, K.J. Healy, D.H. Withers, and B.L. Nelson, 142-149, Association of Computing Machinery, New York, NY.

[12] Fan Zhang, Benxiong Huang: "HLA-Based Network Simulation for Interactive Communication System" in Proceedings of the First Asia International Conference on Modelling & Simulation, 2007.

[13] Sven Pawletta, Wolfgang Drewelow, Thorsten Pawletta. HLA-Based Simulation within an Interactive Engineering Environment. ds-rt, p. 97, Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT'00), 2000.

[14] James O. Henriksen. SLX: The X Is For Extensibility. http://citeseer.ist.psu.edu/518548.html.

[15] U. Klein, S. Straburger, and J. Beikrich. Distributed simulation with JavaGPSS based on the High Level Architecture. In P. A. Fishwick, D. R. C. Hill, and R. Smith, editors, Proceedings of the 1998 International Conference onWeb-Based Modeling and Simulation, pages 85--90, January11-14 1998. San Diego, CA.

[16] Zeigler, B. P., and J. S. Lee. 1998. Theory of quantized systems: Formal basis for DEVS/HLA distributed simulation environment. In Enabling Technology for Simulation Science (II), SPIE AeoroSense 98, Orlando, FL.

[17] Zeigler, B. P., G. Ball., H. J. Cho., J. S. Lee. and H. Sarjoughian. 1999. Implementation of the DEVS formalism over the HLA/RTI: Problems and solutions. Spring Simulation Interoperability Workshop (SIW), (Orlando, FL, March 14--19), 99S--SIW--065.

[18] G. Zacharewicz, C. Frydman, N. Giambiasi. Mapping PIOVRA in GDEVS/HLA Environment. In Proceedings of Summer Simulation Multiconference (SummerSim'07), The society for modeling and simulation international, SCS, San Diego, CA, USA, July 2007.

[19] HLA Toolbox for developing and executing HLA federates from MATLAB. http://www.mathworks.com/products/connections/product_main.html?prod_id=696&prod_name=HLA%20Toolbox.

[20] ForwardSim Inc.: http://www.forwardsim.com/.

[21] MATLAB and Simulink interfaces to HLA and DIS based distributed simulation environments. http://www.mathworks.com/products/connections/product_main.html?prod_id=696&prod_name=HLA%20Toolbox.

[22] MÄK Technologies: http://www.mak.com/.

[23] DEVS and HLA: complementary paradigms for modeling and simulation?. Transactions of the Society for Computer Simulation International, v.17 n.4, p.187-197, Dec. 1, 2000.

[24] Tainchi Lu , Chungnan Lee , Wenyang Hsia , Mingtang Lin. Supporting large-scale distributed simulation using HLA. ACM Transactions on Modeling and Computer Simulation (TOMACS), v.10 n.3, p.268-294, July 2000.

[25] DMSO. HLA Homepage. http://hla.dmso.mil/