C2 Wind Tunnel: Component Integration

H. Neema, Gy. Balogh, G. Karsai, J. Sztipanovits (Vanderbilt/ISIS)
Sep. 13, 2007
Overview

- Basics
  - What is HLA?
  - The C2W Wind Tunnel Component Integration Framework
- Component Integration
  - Java for Scenario Manager Federate (and other simple procedural federates)
    - Scenario Configuration, Control, and Monitoring
  - Simulink models for Dataflow models and Signal Processing
  - DEVSJAVA models for Discrete Event Simulations (DEVS)
  - Colored Petri Nets (CPN) - for Human Organization modeling in the demo
  - OGRE for 3D Terrain modeling
  - OMNET++ network models
- Experimental analysis
  - Configuration of experiments
  - Data collection (federate level and federation level)
  - System performance analysis
The High Level Architecture (HLA)

- An IEEE standard for “interoperable” and “reusable” models and simulations.
  - Most used specification (also used in the demo) is IEEE HLA 1.3 (1998)
  - Most recent specification is IEEE HLA 1516 (2000+)
- DoD-wide policy requires ALL defense models and simulations to comply with the standard.
- Primary goal is to provide a general purpose infrastructure for “distributed” simulation and analysis.
- Software implementing the HLA specification is called Run-Time Infrastructure (RTI).
  - Several commercial and open-source RTIs are available.
  - In the demo we used an open-source RTI PORTICO v0.7.1 implemented in Java language (http://porticoproject.org/).
C2 Wind Tunnel Component Integration using HLA-RTI

Passive Federates
- Data loggers
- Monitors
- Analysis
- Prognostics
- Projections

Live components
- UAVs
- Command & Control
- Live deployment feedback

Simulation Tools
- Simulink
- Omnet
- DEVJSJAVA
- OGRE
- CPN Tools
- Java/C/C++
- etc.
Java Integration (Federate Manager)

**Federation Manager**
- A time-constrained federate
- A time-regulating federate

**Scenario control and configuration:**
- Real time Vs As Fast As
- Possible modes
- Pause/Resume/Terminate Simulation
- Logging level
- Inject debugging/control interactions
- Inject pauses at pre-determined times
- Instrumenting simulations by enabling monitoring of interesting interactions

Injects a “StartScenario” interaction at time 40 for CPN to start analysing the target.

Pauses the entire simulation at predetermined time for describing interesting situations without affecting the simulation execution in any way.

Enables monitoring of the “PosUpdate” interaction which is received from the UAV sensor to plot the UAV trajectory.
Simulink model integration (Vehicle dynamics)

Original model (X4 simulator)

Modified model

Input binding

Add input-output bindings

Output binding

Signal flow

Signal flow

HLA Run-Time Infrastructure (RTI)

GME integration model

Generated .m Receiver and Sender S-function code + .java code for representing Simulink federate

Code generation

RTI runtime communication
DEVSJAVA Discrete Event Simulation (DEVS) model integration

The DEVS Formalism
A Discrete Event System Specification is a structure
\[ M = (X, S, Q, \delta_X, \delta_S, \delta_e, \mathcal{F}, \mathcal{G}) \]

Where
- \( X \) is the set of inputs
- \( S \) is a set of states
- \( Y \) is the set of outputs
- \( \delta_S : S \times X \rightarrow S \) is the external transition function, \( \delta_S : S \times X \rightarrow S \) is the internal transition function
- \( \delta_e : Q \times X \rightarrow S \) is the total transition function, where \( Q = \{(z, e) | e \in X, 0 \leq e \leq s \} \) is the time elapsed since last transition
- \( J : S \rightarrow Y \) is the output function
- \( i : S \rightarrow \mathbb{R} \) is the time advance function

The DEVS Simulation Process
1. Initialize
2. Inject at time \( i \)
3. When receive \( x \)
   - If \( i < N \), then
     - \( i = i + \Delta(i) \)
   - \( i = i + \Delta(i) \)
4. If \( i \geq \Delta(i) \) and \( a \) is message received, generate output \( j(a) \)
5. \( i = i + \Delta(i) \)

The DEVS Coordination Process
- \( S \) is set of states
- \( \mathcal{F} \) is set of transitions
- \( \delta_X, \delta_S, \delta_e \) as defined above

Generated Java code for HLA Bridge (Viewable Atomic model)
Domain-independent (reusable) Java integration classes
HLA Run-Time Infrastructure (RTI)
Image Processing Base DEVSJAVA Model
Modified Model
GME Integration Model

Colored Petri Nets (CPN Tools) model integration (Human organization models)

Modeler can define how the CPN model is connected to the federation.

Generate I/O binding monitoring

C2W Modeling Environment

Import CPN model
- CPN I/O
- CPN types (color sets)

CPN Execution Engine (HLA federate, java)

CPN model
- Loads external CPN model
- Synchronizes CPN model execution with RTI
  - Adds Sim Control place and transition to CPN to ensure model time progress
  - One step rollback optimistic execution (state save/restore)
- Converts HLA interactions and object attribute change events to CPN tokens and back.
- Updates CPN input places with incoming messages
- Reads and removes output tokens and send them as HLA interactions or object state change messages

Maps between HLA messages (interactions and object attribute changes) and CPN input and output places.

Import CPN model
- CPN I/O
- CPN types (color sets)

Generate I/O binding monitoring

HLA Run-Time Infrastructure (RTI)

C2W Modeling Environment
3D Visualization model integration

- **OGRE 3D** (open source graphics engine)
  - Widely used 3D engine in games
  - C++ implementation

- **C2 Wind Tunnel integration**
  - Simple java interface for OGRE (most of our federates are java based)
  - The UAVSensorFed federate: An example visualization federate
    - Interpolation for smooth animation
      - Time interpolation
      - Object position estimation (dead reckoning)
Network simulation integration

- Omnet, Inet packages
  - Omnet is a generic discrete event simulation package (module specification with .ned files, implementation in C++, modular, customizable plugin architecture)
  - Inet: network protocols for omnet (IP, wireless, ad hoc, etc)
- Omnet integration
  - Challenges
    - Scheduler integration
    - Data type mapping
  - C2 Wind Tunnel network support
    - Build in NetworkSim federate, takes care of omnet scheduler synchronization and data conversion
    - Built in network interaction (NetworkInteractions)
    - Derive interactions from the NetworkInteraction to specify custom data types
    - Derived interactions will be sent through the network simulator
    - Federates can be connected to network endpoints, addressing is based on endpoint names
Data collection support

- **C2 Wind Tunnel support on federation level**
  - Fixed vector file format (.vec) for numeric data. Simple, easy to export to other tools (Excel, Matlab)
  - Visualization tool for numeric data (Plove, part of the Omnet package)
  - Built in interactions to support centralized logging
  - Federation Manager GUI shows and logs built in log interactions
  - Monitor specific interactions and convert them to vector files

- **Federate level**
  - Each domain specific simulation environment has its own data collection support
  - Federates can generate log interactions
  - Federates can generate vector files
Experiment configuration and execution

- Configure domain specific simulators
  - Each domain specific simulator has its own configuration files
  - Configuration files are collected to an experiment folder
- C2W support
  - Main configuration file
    - Control the scenario by scheduling interactions
    - Registering monitors
    - Registering pause points
  - Execution script
    - Remote execution of federates
    - Control from one machine through the Federation Manager (pause/resume simulation)
  - Future work
    - Scripting language support (python)

- Demo example configuration
  - cpn model
  - omnet network model
    - omnetpp.ini
    - test.ned
    - tkenv.cfg (gui configuration)
  - run1.xml (main config)
Demo - event flow

Environment | CAOC | Operator | UAV
---|---|---|---
Initial command to CAOC | BDA Request | | Waypoints
| BDA Acknowledgement | | |
| BDA Result | | |
| Response from CAOC | | |
Demo - data flow

UAV vehicle dynamic
50 Hz inner loop

10 Hz position update

UAV
40 Hz sensor output displayed
10 Hz sampling

8Mb/sec signal

CAOC

<<1kb/sec, events

10Mb/sec bottleneck!

2x 8Mb/sec signal

UAV operator
10Hz display

2x 8Mb/sec signal

Image processing
(DEVS Java)
Demo – hardware environment

- Heterogeneous environment
  - Computer 1
    - Windows
    - Simulink, DEVS Java, Omnet (network), Federation Manager
  - Computer 2
    - Windows
    - UAV Sensor
  - Computer 3
    - Windows
    - UAV Operator
  - Computer 3
    - Linux
      - CPN model (human organization)
  - 100Mb dedicated LAN