Using Integrative Modeling for Advanced Heterogeneous System Simulation

Tapasya Patki, Hussain Al-Helal, Jacob Gulotta, Jason Hansen, Jonathan Sprinkle

Thm (Kelly): If all you have is a hammer, look out for your thumb, and all that.

Corollary: If everyone has a hammer, then you will not use anyone else’s nails.

In collaboration with Vanderbilt University, and the University of California, Berkeley. Supported by US AFOSR.
What kinds of heterogeneous systems are we interested in?
What motivates us to integrate?
What domain is this anyway?

• Command and Control
  – Lots of decision makers, looking at tactical data, making command decisions.
  – Lots of monitors
  – Lots of data
  – Lots of decisions to make

• Types?
  – Tactical actors (manned/unmanned components)
  – GUI elements (human interfaces)
  – Vignettes (tactical tests)

• Example vignette:
  – A UAV is sent to a location to look for blue trucks. After a blue truck is spotted, the UAV reports its location. The C2 staff tell that UAV to “track” the blue truck. The UAV then stays as close as it can.

• Example problems:
  – What kind of UAV is it? What kinematic/dynamic properties does it have?
  – What connection/network settings do I use? Am I communicating via TCP/IP, or something more primitive?
Graphically?

* From an unpublished manuscript by Balogh, et al.
What’s the hardest thing about integrative modeling?

The integrative modeling part...

VEHICLE POSITION (JAVA) → Controller and Object Search (MATLAB SIMULINK) → Visualization (DELTA-3D)

Position Information → Controller and Object Search

Position Information

New Waypoint → Controller and Object Search

Search for target

Track target

Found Target

Lost Target

Position Update
Defining the Message and Integration Types

SimpleMessage
message: String

SimulationControl

SimLog
Comment: String
Time: double
FedName: String

SimPause

SimResume

SimEnd

NetworkInteraction

MediumPrio

LowPrio

VeryLowPrio

HighPrio

BestEffortReceive
send interaction handle: int

UAControl
uav_id: String
control: String

BestEffortSend
size: int
sender ip: String
receiver ip: String

InteractionRoot

TextMessage
message: String

PostUpdate
time: double
z: double
y: double
x: double
yaw: double
w: double
v: double
pitch: double
roll: double

UACommand

NewWayPoint
ts: double
ys: double
zs: double
xs: double
yzs: double
ys: double
ws: double

SearchForTarget
feature: String
radius: double
z: double
y: double
x: double

TrackTarget
suspected_z: double
suspected_y: double
suspected_t: double
suspected_r: double
suspected_x: double
vz: double
vy: double
vx: double
target_t: double
target_r: double
target_z: double
target_y: double
target_x: double
target_id: String

ReturnToBase
time: double
Base ID: String

AutomaticBDA
z: double
y: double
x: double

DirectControl
TCD: String

Patki, et al. "Integrative Modeling...Heterogeneous Simulation"
Defining the Message and Integration Types

ObjectRoot

Vehicle

SendStream

ReceiveStream
Defining Component Interconnectivity

Patki, et al. "Integrative Modeling...Heterogeneous Simulation"
Making our UAV look right

A “default” rotorcraft

- A realistic model of the STARMAC was created using Blender
- Rendered model used to represent UAV during simulation
- Multiple instances of the rendered model can be used to simulate swarms of rotorcraft
How do those vehicles fly?
A small wrinkle: image processing in simulation

Delta 3D

Software

Camera Sensor Algorithm

Reported Location

If Object is In FoV

Gives GPS Location of Object

Get a Pixel Location for the Object

Introduce Error to Simulate Data

Object is Now in Some Area

Return to GPS Coordinates

Patki, et al. "Integrative Modeling...Heterogeneous Simulation"
• The GME Paradigm: what gets generated?
• Camera models: where are things executed?
• Vehicle dynamics: where does the logic live?
A thought on domains (again...)

- In the “domain” of heterogeneous simulation
  - Every player has their own domain
  - That domain has the correct tools, etc., for doing development
- There are lots of hard pieces
  - Hard for domain-experts to understand middleware programming
  - Hard for middleware programmers to understand domain concepts
  - Hard for anyone to install everyone else’s tools...
- But, major benefits, if pieces can be integrated easily
  - Allows immediate work on domain-problems, deferring integration work until later
  - Permits domain-specific work to use the tools of the domain
  - Showcases the power of code generation (when used by Jedi appropriately)
Conclusions, and Future Work

- We were able to stand up a significant demo within 3 months of (beginning) to install the software
- Our work concentrated on developing domain-specific pieces to improve visualization and design-time analysis
- The modeling infrastructure supported our development in the appropriate level of abstraction for future integration

Future Work
- More advanced control algorithms (mesh travel/stability)
- More advanced code generation (autogenerate vignette scripts, etc.)
We’re always looking for good graduate students!

http://www.ece.arizona.edu/~sprinkjm/research/c2wt/