Cloud-Based Deployment and Distributed Execution of Models

Rob Kewley, USMA
Motivation

Chain of tools for using M&S to develop system of systems architecture

How to we connect the system design properties to the simulation inputs?
Connect Systems Architecture to Executable Model

Current Approaches

Agent Based Models
Engineering Resilient Systems

Link many independent models
Framework for Assessing Cost and Technology

HLA/DIS
Always On

Frameworks
Orchestrated Simulation through Modeling
We have tried HLA/DIS before...
...HLA/DIS integration does not scale....

- 5 different information exchange protocols
- 4 different terrain databases
- Runs only in real time
- Complex scenario initialization must be manually coordinated across federates
- Scenario execution is manual and error-prone
- Data collection complex and causality can be impossible
- Changing the scenario is a months-long proposition
OneSAF

We need these models

Vehicles

UAV

IWARS

We need these models

Dismount Movement

Dismount Soldier

RTI

FOM
HLA Approach

Simulations have baggage

OneSAF
- OTF
- MSDL
- AAR
- Real Time
- Initialization
- Vehicles
- UAV

IWARS
- Open Flight
- Scenario
- Output Analysis
- Real Time
- Initialization
- Dismount Movement
- Dismount Soldier
Bring your models a “pure” state transition functions. Leave the baggage at home.
Things to Discuss

- Federate models, not simulations
  - Summer vacation at AMSAA

- Loosely managed distributed architecture
  - Models are services via an interface (BOM)
  - Communicate via messaging (Actor Model)
  - Compose models in an interface (...like ProModel)
  - Systems model (SysML) drives model parameters
  - Distributed and parallel execution engine (DEVS-Akka)
  - Support with design and analysis of experiments

- Take advantage of latest advances
  - Enterprise technologies
  - Discrete Event Systems Specification (DEVS) models
  - Actor model of computation

- Proof of principle implementation

- Target implementation for small UAS
Web Based Modeling and Simulation for Analysis

Develop capability to deploy Army models and simulations to distributed users via web a cloud technologies.

Objectives

<table>
<thead>
<tr>
<th>Current Business Model</th>
<th>Proposed Business Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy as software</td>
<td>Deploy as web services</td>
</tr>
<tr>
<td>Accessed locally by a few users</td>
<td>Run distributed by many users</td>
</tr>
<tr>
<td>Data deployed separately</td>
<td>Data integrated and updated</td>
</tr>
<tr>
<td>Single programming language</td>
<td>Cross-platform integration of models in different languages</td>
</tr>
</tbody>
</table>

Deliverables

- AMSAA models deployed on classified network as services with web and simulation interfaces
  - ACQUIRE-TTPM-TAS
  - Direct Fire Accuracy
  - Dismounted Vulnerability
  - Dismounted Mobility
  - Hand Grenade Accuracy
- Proof of principle web-based simulation integrating these models
Acquire Service

Current Configuration

<table>
<thead>
<tr>
<th>Aggregate</th>
<th>Attribute</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>timeOfDay</td>
<td>DAY</td>
</tr>
<tr>
<td>Time</td>
<td>season</td>
<td>SUMMER</td>
</tr>
<tr>
<td>Time</td>
<td>hourOfDay</td>
<td>1300</td>
</tr>
<tr>
<td>Scene</td>
<td>clutter</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>Scene</td>
<td>background</td>
<td>VEGETATION</td>
</tr>
<tr>
<td>Scene</td>
<td>lightLevel</td>
<td>1</td>
</tr>
<tr>
<td>Scene</td>
<td>region</td>
<td>URBAN</td>
</tr>
<tr>
<td>Weather</td>
<td>weatherCondition</td>
<td>15</td>
</tr>
<tr>
<td>Weather</td>
<td>massExtinctionCoefficient</td>
<td>0</td>
</tr>
<tr>
<td>Weather</td>
<td>attenuationCoefficient</td>
<td>0.326</td>
</tr>
<tr>
<td>Weather</td>
<td>obscurantConcentrationLength</td>
<td>0</td>
</tr>
<tr>
<td>Weather</td>
<td>rPath</td>
<td>0</td>
</tr>
<tr>
<td>Weather</td>
<td>meteorologicalVisibilityRange</td>
<td>12</td>
</tr>
</tbody>
</table>

Service Control

Current Configuration

Turn Debugging On/Off

Play/Pause Service

Start/Stop Service

Messages
Acquire Test Client

Message
- Observer: [Name]
- Name: MIA2_DVO
- Type: [Type]
- TFOV uRand1: [Value]
- TFOV uRand2: [Value]

Response
- Observer: MIA2_DVO
- Target: RED1
- Acquisition Level: DETECTION
- Correct ID: false
- TFOV: [Value]

Statistics
- Request sent at: May 27, 2015 at 1:52:10 PM EDT
- Response received at: May 27, 2015 at 1:52:10 PM EDT

Elapsed Time: 0.015s

Log Messages:
- Connection to server: [Details]
- Reason: The connection was closed abnormally, e.g., without sending or receiving a Close control frame
Enterprise Model Integration

- Design and Analysis of Experiments
- Enterprise Architecture
- Simulation Execution Engine
- Enterprise Data and Services
- Messaging System
- Enterprise Data Model
  - Model A
  - Model B
  - Model C
  - Model D
EASE-DMF provides the tools for the simulation analyst and engineer to build scenarios, combine models, set properties, and collect data in order to analyze system performance.

System models are functional, discrete, and modular. They can be added as needed to the simulation and run in parallel. In contrast to current federations, the analyst combines many simple and functional models of individual systems, not entire simulations.
Useful Theories - Simulation

- **Discrete Event Specification (DEVS)**
  - DEVS models are modular
  - Composable hierarchies in coupled models
  - Strong track record

- **Base Object Model (BOM)**
  - Adds semantics to the models
  - Complete specification of data inputs and outputs
  - Situates models in a chain or interactions

- **Parallel algorithms**
  - Optimistic time advance – Time Warp
  - Supports distributed and cloud-based implementations
Useful Theories – Computer Science

- Functional programming
  - Functions are composable
  - Functions have no side effects
  - Predictable behavior

- Actor model of computation
  - Encapsulation of state
  - Responds to messages by...
    - Sending messages to other actors
    - Changing state in a way that influences future messages
    - Creating new actors

- Reactive programming
  - Event driven
  - Responsive
  - Asynchronous
  - Loosely coupled messaging
  - Fault tolerant
Implementation of reactive actor model
- Scala and Java versions
- Each DEVS model is an Akka SimActor
- Each actor runs on its own thread
- Exchanges data only through messaging
- Support for serialization and distributed actors
- Ability to manage threads
- Open source
ACQURIE Sensor Model

- **Soldier DEVS Model**
- **Awareness DEVS Model**
- **Sensor DEVS Model**
  - **Properties**
    - Sensor Properties
    - Target Properties
    - Weather
  - **State: Sensor Orientation & Location**
  - **Schedule**
  - **Slew Sensor** (internal State Transition)
  - **Event Handler**
    - **Acquire** (External State Transition)
      - Acquire Inputs
      - Acquire Outputs
  - **Mappings**
    - Target Detection
    - Target Detection Time
    - Background Exposed Area
  - **Terrain Model**
  - **Location Time**
  - **Target Location**
Exposing Function Interfaces

- **As a class in API**
  - Use static class
  - Pass in initialization data at instantiation
  - Use public static methods

- **As a state transition function service**
  - Define message classes for input/output
  - Map message classes to messaging protocol such as:
    - JSON
    - Protocol Buffers
    - XML
  - Expose through a service interface
  - Document everything thoroughly

```java
public final class DeadReckon {
    private static Velocity velocity;
    public DeadReckon(VehicleType t) {
        velocity = getVehicleVelocity(t);
    }
    public static Location newLocation(Location l, int time) {
        return new Location(l + velocity * time);
    }
}
```
Composing DEVSActors

- As DEVSActor
  - Akka or C++ Actor Framework (CAF)
  - Pass in initialization data at instantiation
  - Define message interface

- As a distributed actor
  - Map message classes to messaging protocol such as:
    - JSON
    - Protocol Buffers
    - XML
  - Akka or CAF remote actors
  - Document everything thoroughly

- As simulation
  - Compile tightly linked models in a coupled actor
  - Compile models with heavy interaction traffic in same JVM (Akka) or native library (CAF)
  - Control with execution engine
  - Copy large interactive data sets such as terrain to each process to save network traffic
EASE-DMF Use Cases

Simulation Development

- Map elements of systems architecture to models
- Define performance measures
- Construct simulation
- Verify simulation model
- Validate simulation model
Small Unmanned Aircraft Modeling Framework

Summary

- New SUAS acquisition strategy requires PM UAS to operate as lead systems integrator (LSI)
- LSI role requires assessment of engineering trades
- Develop a web-based integrated modeling and simulation framework to assess engineering trades for small unmanned aircraft systems

Technical Approach

- Develop a series of discrete analysis models to represent small UAS technical performance
  - Target acquisition
  - Communications propagation
  - Situation awareness
  - Power
  - Flight dynamics
  - Terrain

- Wrap in web-based interface
- Integrate in simulation framework using Discrete Event Systems Specification (DEVS) and SysML

Deliverables

- SysML architecture of proposed micro UAS
- Scenario development and design of experiments
- Simulation analysis of micro key UAS performance parameters tradespace
  - Size, weight, and power
  - Fixed vs. Rotary wing
  - Noise
  - Sensor performance
  - Radio performance
- Squad or platoon level mission performance metrics
Functions of a DEVSActor

- Initialize with static properties
- Local and global virtual time
- An internal schedule of state transitions
- A set of internal and external state transition functions
- It may update its state, schedule and internal event, or generate a message to other DEVSActors
- Maintain record of internal state and drop as global time advances
- List of events that may be replayed in the event of roll back
- An ability to generate random variates needed by state transitions
- A publish and subscribe mechanism that asks for a notification message upon specific state transitions of other DEVSActors
M&S Composition
State Transition Functions

- Rules for functions:
  - Must be stateless
  - May use initialization data under closure
  - These are static objects with only static data and methods
  - After initialization, calling with same data always gives same answer

Functions are mathematically composable

State transitions that take place at same instant can be chained into one state transition

Calls out to stateless utility functions such as coordinate transformation