A Coordinated Initialization Process for the Distributed Space Exploration Simulation

2007 Spring SIW

Robert G. Phillips, Dan Dexter, David Hasan / L-3
Edwin Z. Crues, PhD / NASA

Introduction

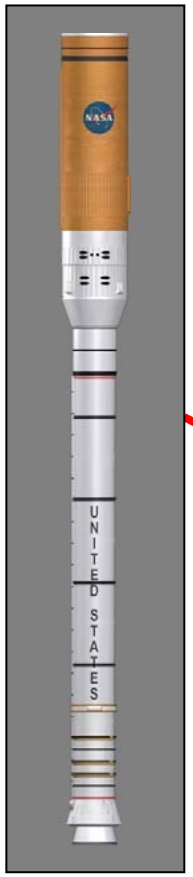
- Describe the federate initialization process developed for the Distributed Space Exploration Simulation (DSES)
- Based upon the HIIA Transfer Vehicle (HTV) Distributed Flight Controller Trainer
- Uses IEEE 1516 High Level Architecture
  - Original paper was “how to” guide for developers new to HLA

Background: DSES

- Supports NASA's Crew Exploration Vehicle (CEV) and Constellation programs
- The NASA centers providing Federates representing primary vehicle element areas are as follows
  - Johnson Space Center: CEV Command and Service Modules
  - Marshall Space Flight Center: Crew Launch Vehicle (CLV) Stages 1 and 2
  - Langley Research Center: Launch Abort System (LAS)
  - Ames Research Center: Crew
- Model joint operations in real time
  - Vehicle launch
  - Separation
  - Other joint operations

DSES

MSFC



JSC

ARC

Integrated Distributed
Orion/Ares Simulation

LaRC

Simulation requirements

- Need to test many possible missions and mission phases with multiple vehicle configurations
  - Dynamic set of vehicles (federates)
  - Dynamic exchange of initial data
  - No particular federate start order

Nine Step Initialization

1. Create the federation
2. Publish and subscribe
3. Create object instances
4. Confirm all federates are joined
5. Achieve “initialize” Synchronization Point
6. Update object instance(s) with initial data
7. Wait for object instance reflections
8. Set up time management
9. Achieve “startup” Synchronization Point

Step 1: Create the Federation

- Each federate attempts to create the federation
- Each federate joins the federation
  - Federate is named for represented NASA center, e.g., “JSC”

Step 2: Publish and Subscribe

- Each federate publishes and subscribes to all required objects and interactions
  - In other simulations, it would be sufficient to publish and subscribe to all objects and interactions necessary to exchange initial data

Step 3: Create Object Instances

- All object instances are created
  - Federation-unique names are assigned to instances with standardized naming conventions
    - E.g., federate_name.object_name.instance_name
  - Again, only instances necessary for initial data exchange is strictly necessary at this point

Step 4: Confirm All Federates Have Joined

- Each federate is configured with a list of required federates for the run
  - This list is the only necessary pre-run data exchanged between centers
- Federates use the Management Object Model (MOM) interface to confirm that all required federates have joined
  - We use successful creation of “initialize” Synchronization Point to dynamically determine a *master federate* that performs this check

Step 5: Achieve "initialize" Synchronization Point

- Each federate achieves the "initialize" Synchronization Point to indicate that it is ready to exchange initial data
  - All necessary objects and interactions have been published and subscribed
  - All necessary object instances have been registered and discovered
  - At least one (master) federate waits until all required federates are joined

Step 6: Update Object Instances With Initial Data

- Each federate enables asynchronous delivery
- Each federate sends object attribute updates with initial data in Receive Order (RO)
  - Same instances are used for initial RO updates that are used for run time Time Stamp Order (TSO) updates

Step 7: Wait for Object Reflections

- Each federate waits until required object instance reflections are received
  - Federates need to dynamically understand what reflections are needed based upon list of required federates

Step 8: Set Up Time Management

- Set up Time Management as required
  - Time Regulating
  - Time Constrained
  - Disable Asynchronous Delivery

Step 9: Achieve "startup" Synchronization Point

- Each federate achieves the "startup" Synchronization Point to indicate that it is fully initialized and ready to execute
- When the federation is synchronized, the simulation begins execution

Conclusions

- An understanding of the IEEE 1516 standards and the FOM/SOMs for a simulation are not sufficient for a robust initialization process
- We have shown an example of a uniform robust initialization process shared by all federates that
  - Ensures that all necessary federates are joined before the simulation starts
  - Allows the set of necessary federates to be easily modified
  - Doesn't require a specific federate start sequence
  - Allows dynamic exchange of initial object attribute values during startup