Design and Principles Enabling the Space Reference FOM

2017-SIW-038

Björn Möller, Pitch, Sweden
Edwin Z. Crues, NASA, USA
Dan Dexter, NASA, USA
Alfredo Garro, University of Calabria, Italy
Michael Madden, NASA, USA
Anton Skuratovskiy, RusBITech, Russia
Background and Status

• Previous papers present why distributed simulation is important to the Space domain
  ▪ SISO paper 16F-SIW-017 “A First Look at the Upcoming SISO Space Reference FOM” and two ACM/IEEE DS-RT papers

• HLA federations for the Space domain have been developed for almost twenty years
  ▪ There are plenty of experiences to collect and reuse

• Two years ago the SISO Space Reference FOM PDG kicked off

• The first complete draft is now available

• Next step: review and testing
Federate Roles and Types of Design Patterns

• This paper presents important design patterns used in the Space FOM
  ▪ Essential versions. Read the draft standard for details

• Three key federate roles are used:
  ▪ The Master role
  ▪ The Pacer role
  ▪ The Root Reference Frame Publisher role

• Three types of patterns are covered in this paper:
  ▪ Execution Control design patterns
  ▪ Time Management design patterns
  ▪ Spatial design patterns
Execution Control Design Patterns

• These patterns are mainly used to allow the Master to manage the flow of execution

• The patterns are:
  - Removal of orphaned federation execution *
  - Centralized checking of required federates
  - Detection if a federate is a late joiner
  - Global configuration data in singleton instance
  - Synchronized multi-phase initialization
  - Central execution control with transition requests

*) Not presented here – read the paper
Centralized checking of required federates

• A certain set of federates need to be present before the simulation can start.
  - This may be for technical reasons, or to be able to perform a meaningful simulation.

• The Master knows which federates are required and performs the check

• Each individual federate may not know if it is required or not
Centralized checking of required federates

- Master Federate Starts
  - Check joined federates using MOM
  - Required federates joined
    - Yes: Register Sync Point "Initialization Started"
    - No: End

- Other Federate Starts
  - Synch Point "Initialization Started" registered
    - Yes: End
    - No: End
Detection if a federate is a late joiner

• This pattern applies to a federate that may execute as either an early joiner or a late joiner. Late joiner means that the initialization has already been completed.
  ▪ In case it joins a federation early, it needs to complete certain initialization steps, potentially in coordination with other federates.
  ▪ In case it joins late, different steps may need to be performed.
• This pattern extends upon the previous pattern.
Detection if a federate is a late joiner

Master Federate Start

Create & Join

Perform Master Initialization

Achieve Sync Point “Initialization Started”

Register Sync Point “Initialization Completed”

End

Other Federate Start

Synch Point “Initialization Started” registered

No

Yes

Perform Early Joiner Initialization

Achieve Sync Point “Initialization Started”

End

Perform Late Joiner Initialization

End

Synch Point “Initialization Completed” registered

No

Yes
Synchronized multi-phase initialization

- Before starting the main execution, federates need to exchange initial data. Some of the data cannot be calculated before some other data has been provided by some other federate.

- To be able to control and verify that all data has been provided, the federation needs to go through a specified set of initialization phases.
  - In this example “Phase A” and “Phase B”
  - Practical example: initialize a multi-stage rocket
Synchronized multi-phase initialization

- Master Start
  - Register Sync Points “Phase A” and “Phase B”
  - Achieve Sync Point “Phase A”
  - Wait for “Phase A” synchronized
  - Achieve Sync Point “Phase B”
  - Wait for “Phase B” synchronized
  - End

- Non-Master Start
  - Send any data for Phase A
  - Wait for required data for Phase A
  - Achieve Sync Point “Phase A”
  - Wait for “Phase A” synchronized
  - Next phase
Global configuration data in singleton instance

- A federation needs to share a number of global properties
  - Storing static data in configuration files for each federate introduces a risk of mismatching data.

- Sample static data:
  - Epoch (start time)
  - References to important object instances

- Sample dynamic data:
  - Execution state

- In this case the Master is responsible for sharing the data
Global configuration data in singleton instance

Execution Configuration Object
- Epoch: 01-Jan-2017 00:00
- RootRefFrame: MoonCentricInertial
- RunMode: Running
- NextMode: Freeze
- NextModeTime: 12345.66

Master Federate
Federate
Federate

Runtime Infrastructure – RTI
Central execution control with transition requests

- Federates need to transition between initializing mode, running mode, freeze mode and shutdown in a controlled manner.
- Any federate may need to request a mode transition.
- Since federates may use different time steps, or may need some time to transition, the transition may not happen immediately.
- Late joining federates must perform a required transition, even if the transition was requested before a federate joined.
Central execution control with transition requests

No sync for Shutdown!
Time Management Design Patterns

• These patterns coordinate the advance of time, exchange of time stamped data and synchronization with physical time
  ▪ Physical time or “real world time” in the Space Reference FOM is based on the classical Newtonian concept of absolute time, which is a simplification compared to the relativistic space-time concept.

• Closely related to the execution control patterns

• The patterns are:
  ▪ Constant but potentially different federate time steps
  ▪ Mix of paced scenario time and physical time
Constant but potentially different federate time steps

- A number of federates that use time-stepped simulation need to execute together in a federation.
- The time-steps are constant but may be different between federates.
- Internally, each federate has a native time step for the physics model.
- The federation needs to have well-defined points in time when the federation wide state is complete and consistent, for example for check-pointing, snap-shooting or freeze of the federation.
**Constant but potentially different federate time steps**

- **Federation Time Step =** Pacing time step
- **Federate Time Step =** Time step used for time advance by a federate.
  - Shall be \( n \times \) Federation Time Step where \( n \geq 1 \)
  - Shall be \( n \times \) Simulation Time Step where \( n \geq 1 \)
- **Simulation Time Step =** native time step of internal physics model of a federate.
- **Freeze may occur at Common Time Boundaries**
Mix of paced scenario time and physical time

• An HLA federation can accommodate both simulations running in soft real-time and simulators that use central timing equipment (CTE) (e.g., a GPS timing board) for hard real-time synchronization.

• The HLA federation is capable of going to freeze, and later resume.

• The simulations that synchronize using the CTE, must also be able to handle these mode transitions.
Mix of paced scenario time and physical time

- The HLA Logical time line and the CTE physical time line are connected during Run mode
  - An offset is calculated when entering Run mode
- When entering Freeze mode they are disconnected

CTE Time Line

12:01:05  12:01:06

Calculate Offset

0 10 20

Scenario Time Line

Run Freeze Run
Synchronizing CTE and Logical Time

1. Go to Run mode
2. Calculate Scenario time to CTE offset
3. Check that Time Advance Granted has been received
   - Simulate this time step
   - Time Advance Request
   - Wait for next CTE time
4. Mode
   - Freeze, Shutdown
   - Other Modes
   - Run
Spatial Design Patterns

• **Space simulations may include assets that operate on or about celestial bodies other than the Earth.**
  - There is no common reference frame of convenience for all space simulations.

• **When modeling operations that span multiple celestial bodies, each federate may prefer to operate an asset in a local reference frame but the federation must relate those reference frames to each other.**

• **The patterns are:**
  - Reference Frames explicitly specified using object instances
  - Replaceable and Extendable Tree of Reference Frames
Reference Frames explicitly specified using object instances

- It is conceptually and computationally inconvenient to perform all calculations using the same coordinate system.
  - Considering the vastness of Space, attempting to use a single coordinate system would introduce unacceptable mathematical rounding errors.
- The solution is to create one object instance of the Reference Frame class for each reference frame that is required.
- Each Reference Frame is identified using a name.
- Positions, for examples for a space vehicle, are given in relation to a named reference frame.
Reference Frames explicitly specified using object instances

- **HLAobjectRoot**

- **ReferenceFrame**
  - name : HLAunicodeString
  - parent_name : HLAunicodeString
  - state : SpaceTimeCoordinateState

- **SpaceTimeCoordinateState**
  - time : Time
  - translational_state : ReferenceFrameTranslation
  - rotational_state : ReferenceFrameRotation

- **ReferenceFrameTranslation**
  - position : PositionVector
  - velocity : VelocityVector

- **ReferenceFrameRotation**
  - attitude_quaternion : AttitudeQuaternion
  - angular_velocity : AngularVelocityVector
Replaceable and Extendable Tree of Reference Frames

• Need to translate coordinates between several different reference frames in order to determine spatial relationships between entities using different coordinate systems.
• Need to be able to switch between different reference frames during execution, for most convenient computations.
• Need to be able to use different sets of reference frames for different scenarios.
• Need to extend common and standardized reference frames with custom reference frames.
Replaceable and Extendable Tree of Reference Frames

- Structure the reference frames into one single directed acyclic graph (i.e. a tree).
- Each reference frame specifies its translational and rotational states with respect to the parent reference frame, except for the root.

![Diagram of reference frames]

- SolarSystemBarycentricInertial
  - SunCentricInertial
  - EarthMJ2000Eq
  - MarsCentricInertial
    - EarthFixed
    - MoonCentricInertial
    - MarsFixed
    - MoonFixed
Comparison with RPR FOM

- **Space FOM**
  - Reliable data exchange
  - Causality and repeatability
  - Well-managed set of federates
  - Coordinated execution with initialization, execution, freeze and shutdown
  - Hard real-time, soft real-time, scaled real-time or as-fast-as-possible
  - Multiple reference frames, standardized or custom
  - Few, generic object classes

- **RPR FOM**
  - Best effort data exchange
  - Not repeatable
  - Ad-hoc set of federates
  - Coordinated freeze/run
  - Soft real-time
  - Earth-centric and entity-centric coordinates
  - Wide range of specialized object classes
  - Wide range of enumerations for entity types, etc
  - DIS compatibility
Conclusion

• The focus of the first version of the SISO Space Reference FOM is execution control, time management, coordinate systems, well-known reference frames, and physical entities

• A number of design patterns and principles for this have been presented

• They are based on many man-years of practical federation development in the Space domain

• They are also suitable for reuse in other domains

• We have a continued interest in exchanging ideas with other simulation domains through SISO