Design and Principles Enabling the Space Reference FOM

Björn Möller
Pitch Technologies
Repslagaregatan 25
58222 Linköping, Sweden
+46 13 4705503
bjorn.moller@pitch.se

Edwin Z. Crues
Simulation and Graphics Branch (ER7)
Software, Robotics, and Simulation Division (ER)
NASA Johnson Space Center
2101 NASA Road 1, Houston, TX
edwin.z.crues@nasa.gov

Dan Dexter
Simulation and Graphics Branch (ER7)
Software, Robotics, and Simulation Division (ER)
NASA Johnson Space Center
2101 NASA Road 1, Houston, TX
daniel.e.dexter@nasa.gov

Alfredo Garro
University of Calabria
Department of Informatics, Modeling, Electronics and Systems Engineering (DIMES)
University of Calabria
Via P. Bucci 41C, 87036 Rende (CS), Italy
alfredo.garro@dimes.unical.it

Michael Madden
Simulation Development and Analysis Branch (D107)
NASA Langley Research Center
24 West Taylor Street, Hampton, VA
Michael.M.Madden@nasa.gov

Anton Skuratovskiy
RusBITech
Varshavskoye shosse 26
117105 Moscow, Russia
+7 495 648 0640
a.skuratovskiy@rusbitech.ru

Keywords: Space simulation, Reference FOM, HLA, Time Management,
Reference Frame, Execution Control, Simulation Initialization

ABSTRACT: *A first complete draft of the Simulation Interoperability Standards Organization (SISO) Space Reference Federation Object Model (FOM) has now been produced. This paper provides some insights into its capabilities and discusses the opportunity for reuse in other domains.*

The focus of this first version of the standard is execution control, time management and coordinate systems, well-known reference frames, as well as some basic support for physical entities. The biggest part of the execution control is the coordinated start-up process. This process contains a number of steps, including checking of required federates, handling of early versus late joiners, sharing of federation wide configuration data and multi-phase initialization.

An additional part of Execution Control is the coordinated and synchronized transition between Run mode, Freeze mode and Shutdown. For time management, several time lines are defined, including real-time, scenario time, High Level Architecture (HLA) logical time and physical time. A strategy for mixing simulations that use different time steps is introduced, as well as an approach for finding common boundaries for fully synchronized freeze.

For describing spatial information, a mechanism with a set of reference frames is specified. Each reference frame has a position and orientation related to a parent reference frame. This makes it possible for federates to perform calculations in reference frames that are convenient to them. An operation on the Moon can be performed using lunar coordinates whereas an operation on Earth can be performed using Earth coordinates. At the same time, coordinates in one reference frame have an unambiguous relationship to a coordinate in another reference frame.

While the Space Reference FOM is originally being developed for Space operations, the authors believe that many parts of it can be reused for any simulation that has a focus on physical processes with one or more coordinate systems, and require high fidelity and repeatability.

1. Introduction

Previous papers [1,2] provide a background on the importance of distributed simulation for the space domain, as well as examples of a number of High Level architecture (HLA) [3] federations in the Space domain, from the late 90's up till today.

Continued discussions in the Simulation Interoperability Standards Organization (SISO) space simulation community led to the initiative to start up a group to develop a standard that could serve as a foundation for better a priori interoperability between space simulations.

A product nomination for a Space Reference Federation Object Model (FOM) was composed, submitted to the SISO Standards Activities Committee (SAC), and ultimately approved by the SAC for development. A Product Development Group (PDG) was formed to develop the Space Reference FOM standard.

1.1. Progress of the standards development

The initial Space Reference FOM PDG meeting was held at the SISO Fall 2015 Simulation Innovation Workshop (SIW) in September 2015. Since this initial PDG meeting, more than 60 meetings have been held. Space Reference FOM PDG members include government, industry and academia. The group has provided draft versions of the standard to the international university outreach program Simulation Exploration Experience (SEE, previously SISO "Smackdown") [4].

As of September 2017, a first, almost complete draft exists. The next step is test and verification of the draft. Three different teams in the drafting group are developing three different implementations of the core components of the Space Reference FOM, which will then be cross-tested in different combinations. After the tests, the standard will go through formal balloting, according the SISO Balloted Products Development & Support Process (BPDSP) [5]. The target is to release the final standard in 2018.

1.2. The standard broken down into design patterns

The standard provides detailed solutions to a number of key requirements in a space federation. In this paper, these solutions are presented in a generic form. This makes it easier to explain their essence.

The approach chosen is known as "Design patterns" [6] in the software community. A design pattern can be described as a "general reusable solution to a commonly occurring problem within a given context" (Wikipedia).

1.3. Federate Roles

Several of these patterns depend on the three federate roles that are defined in the Space Reference FOM. They are:

1. **The Master role**, that controls the initialization and execution of the federation
2. **The Pacer role**, that manages the advancement of scenario time in relationship to the real time
3. **The Root Reference Frame Publisher role**, that publishes the root of the tree of reference frames, as described later in this paper.

The patterns are grouped into three types:

1. **Execution Control** design patterns, where the federates and the Master are important
2. **Time Management** design patterns, where the Pacer and the Master are important.
3. **Spatial** design patterns, where the Root Reference Frame Publisher and the Master are important

All diagrams in this paper are based on the June 2017 draft version of the Space Reference FOM [7], which contains diagrams with considerably more technical detail. These solutions are based on lessons learned from real life federations. Many of them are derived from the Integrated Mission Simulation document [8] by NASA.

2. Execution Control Design Patterns

This section describes the six main patterns that are used for initialization and execution control in the Space Reference FOM. They are described in the order that they are typically executed in a federate.

The patterns are:

1. Removal of orphaned Federation Execution
2. Centralized checking of required federates
3. Detection if a federate is a late joiner
4. Global configuration data in singleton instance
5. Synchronized multi-phase initialization
6. Central execution control with transition requests

Since there are many federates executing in parallel, these patterns may be running in parallel in a federation execution. As an example, several federates may be performing a synchronized multi-phase initialization, while a new federate detects if it is a late joiner or not.

2.1. Removal of orphaned Federation Execution

Requirement: A federation needs to ensure that it executes in a clean federation execution when it starts. If not, the federation execution may contain orphaned object instances, or may have been advanced in HLA logical time. Such a federation execution typically exists if federates in a previous execution did not shut down correctly.

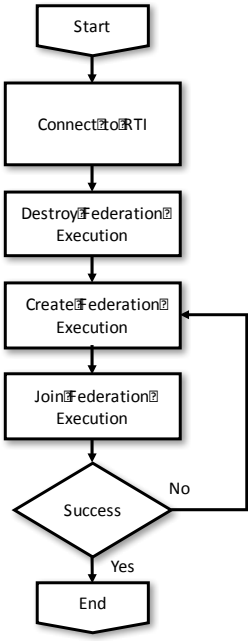

Figure 1: Removal of Orphaned Federation

Solution: After connecting to the RTI, a federate immediately destroys the federation execution. It then creates and joins the federation execution. In case the federation doesn't exist anymore, since another federate just destroyed it, the federate needs to go back and try to create it.

Discussion: Note how this pattern needs to handle the case when several federates are trying to join at almost the same time and may potentially destroy each other's federation executions. This pattern also relies on the property that a federation execution cannot be destroyed, once a federate has successfully joined it.

2.2. Centralized checking of required federates

Requirement: 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.

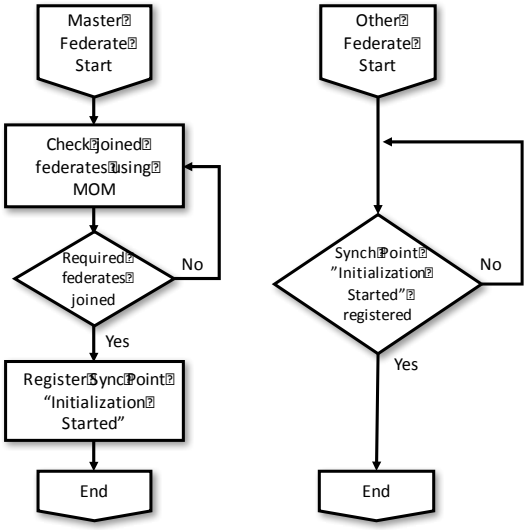

Figure 2: Check for required Federates

Solution: a designated federate is used, in this case the federate with the Master role. It has access to a list of the federate names of the required federates, for example in a configuration file. After joining the federation execution, it uses the HLA Management Object Model (MOM) to monitor which federates that have joined. When all required federates have joined, it registers the synchronization point "Initialization Started".

Any other federate, that may potentially be a required federate, needs to perform the following sequence. After joining, it checks for the synchronization point "Initialization Started". If this synchronization point hasn't been announced, the federate enters a wait loop, where it periodically checks if "Initialization Started" has been announced. When it has been announced the federate will start the main initialization.

Note that this pattern is extended later in the next pattern.

Discussion: In this pattern, the availability of a synchronization point is used as a global flag. The pattern doesn't require any particular start order between the Master federate and the required federates. The pattern will, to some degree, ensure that all required federates start the initialization process at the same time.

2.3. Detection if a federate is a late joiner

Requirement: This pattern applies to a federate that may execute as either an early joiner or a late joiner. Late joiner, in this context, means that the initialization has already been completed. In case it joins a federation early it needs to complete certain initialization steps. In case it joins late, different steps may need to be performed.

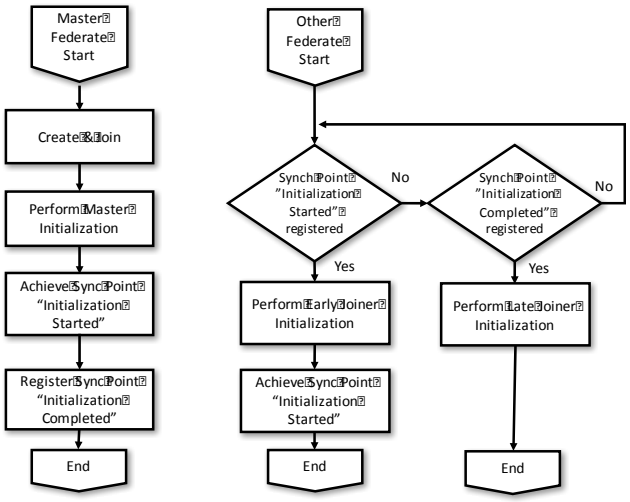

Figure 3: Detection if a federate is a late joiner

Solution: A designated federate, in this case the federate with the Master role, registers the synchronization point "Initialization Completed". If this synchronization point hasn't been announced, the federate will act as an early joiner and go through the initialization steps. Otherwise it will act as a late joiner and will go through the late joiner steps.

Discussion: This pattern also uses the availability of a synchronization point as a global flag. Note that this pattern doesn't guarantee that an early joiner federate enters the initialization steps in sync with other federates.

2.4. Global configuration data in singleton instance

Requirement: A federation needs to share a number of global properties, for example static data, such as epoch or references to important object instances or dynamic data, such as execution state. Storing static data in configuration files for each federate, introduces a risk of mismatching data.

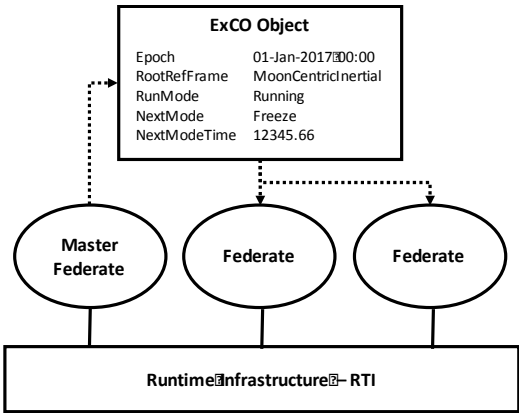

Figure 4: Shared configuration data in singleton

Solution: A dedicated federate registers an object instance of a particular object class with a specific HLA object instance name. The dedicated federate sets the attribute values. For static data, this may be based on configuration data provided to the dedicated federate, or by discovering data in the federation. For dynamic data, other federates may send interactions to provide or request data, as shown later in this paper. Other federates will get the configuration data by subscribing to the particular object class.

Discussion: In the Space Reference FOM, the federate with the Master role registers an object instance called "ExCO", which stands for Execution Configuration Object. It contains information like the Epoch, current run/freeze mode, and name of the root reference frame.

2.5. Synchronized multi-phase initialization

Requirement: 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.

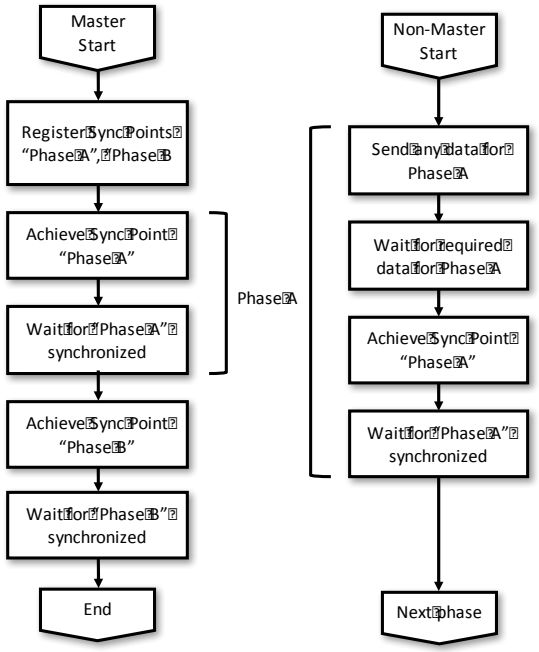

Figure 5: Multi-phase Initialization

Solution: A number of named phases have been agreed upon in advance, each phase with corresponding named synchronization point. In the example in Figure 5 there are two phases called A and B. A dedicated federate role registers these synchronization points. It then achieves them, one by one. After achieving a synchronization

point, it waits for the federation to be synchronized, before achieving the next synchronization point.

Participating federates will perform the following for each phase: first send out initialization data, then achieve the synchronization point and finally wait for the federation being synchronized.

Discussion: In the Space Reference FOM, the Master federate manages the multi-phase initialization. One advantage of this pattern is that it makes it easier to verify and potentially troubleshoot the initialization.

2.6. Central execution control with transition requests

Requirement: 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.

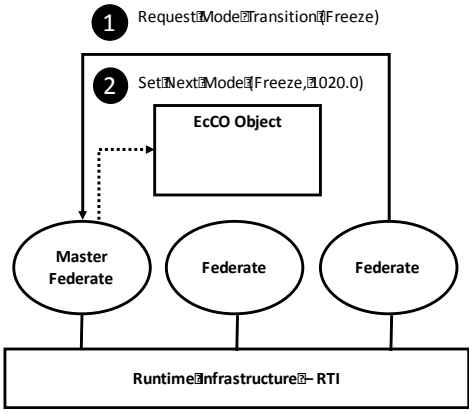

Figure 6: Requesting mode transitions

Solution: A global object instance, in this case the Execution Configuration Object (ExCO), stores the current mode, as well as the next mode, together with the time for the next mode. Any federate can make requests for mode transitions, as shown in Figure 6. The Master federate will calculate an acceptable time for the mode transition and store this in the ExCO.

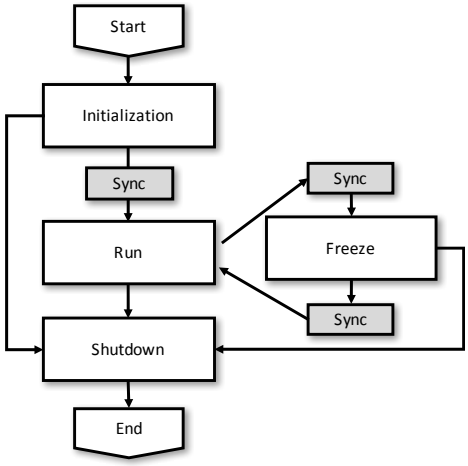

Figure 7: Execution Modes and synchronization

Mode transitions to Freeze or Run start with achieving a synchronization point, in order to synchronize federates that take different time to transition, as shown in Figure 7. Mode transitions to Shutdown do not use a synchronization point. Note that it is possible for the federation to go directly to shutdown, if a federate fails during initialization.

All federates that produce data or have HLA Time Regulation turned on, must transition to the next state as specified in the ExCO. Data loggers and visualizers may not always take part in the state transitions.

Interaction and Attribute Updates related to requesting and performing the state changes need to be sent using Receive Order in a federation using HLA Time Management

Discussion: Transitioning to shutdown needs special consideration in this pattern. An operator may require going to shutdown at any point in time, for example when a federate becomes unresponsive or faulty in other ways. A synchronization point cannot be used in this case, since unresponsive federates may never achieve a synchronization point, thus preventing the entire federation from shutting down.

3. Time Management Design Patterns

This section describes two patterns for managing time. The Space Reference FOM describes several time concepts, where some of the most important are:

Scenario Time is the conceptual time associated with the physical systems that are modeled in the federates.

HLA Logical Time is the time used by HLA to time-stamp and order messages and to regulate time advance. The HLA logical time starts at zero. It can be related to the scenario time by providing an Epoch (starting point).

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.

The patterns are:

1. Constant but potentially different federate time steps
2. Mix of paced scenario time and physical time

The time management patterns are closely related to the execution control patterns, in particular the transition requests between the Run, Freeze and Shutdown modes. Here they are presented standalone, but to get the exact details, the reader is encouraged to read the Space Reference FOM.

3.1. Constant but potentially different federate time steps

Requirement: 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. 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.

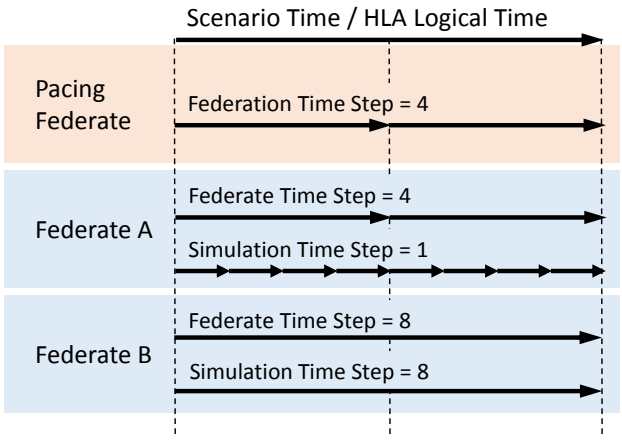

Figure 8: Federate and Federation time steps

Description: A common Federation Time Step is agreed upon. The federate with the pacing role shall advance time using this time step, as shown in Figure 8. Any other federate shall advance time using a time step, called the Federate Time Step, that shall be an integer multiple n>=1 of the Federation Time Step.

Each federate has a native time step of its internal physics model, here called the Simulation Time Step. The Federate Time Step shall be an integer multiple n>=1 of the Simulation Time Step.

The pattern guarantees that there will be repeated HLA Logical times to which all federates will be granted, here called Common Time Boundaries. These can be calculated as the least common denominator of all Federate Time Steps.

Discussion: Many, but not all, physics simulations have configurable time steps, which facilitates the choice of federation time step. If one federate is less flexible in the choice of time step, this may strongly influence the choice of time step. The more important aspect, when selecting time steps for physical models, may be the resolution and fidelity that is required for a particular simulation purpose.

3.2. Mix of paced scenario time and physical time

Requirement: 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. While the HLA federation is capable of going to freeze, and restarting, the simulations that synchronize using the CTE, must be able to handle these mode transitions.

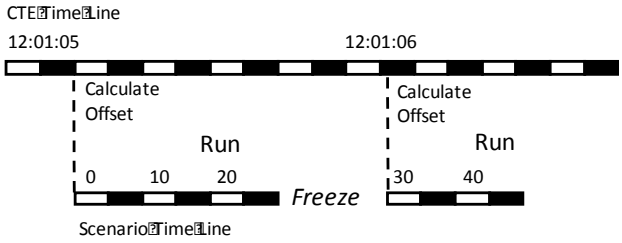

Figure 9: CTE time line and Scenario time line

Description: The federation is regarded as having two time lines, the scenario time-line and the CTE time-line as shown in Figure 9. These are connected in Run mode but disconnected at Freeze mode. The federate with the Master is responsible for connecting them when entering Run mode using a separate CTE epoch, which specifies the offset between the CTE time and the scenario time.

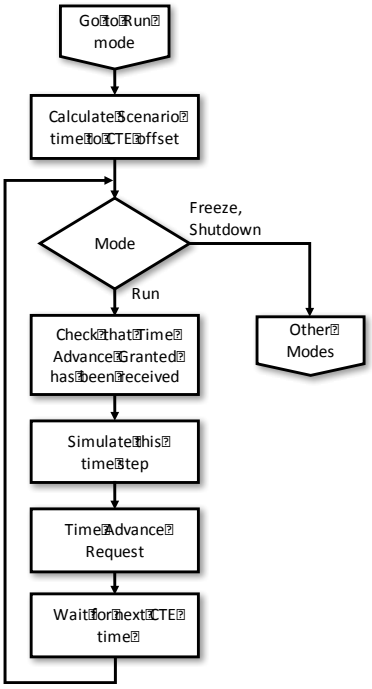

Figure 10: Advancing scenario time vs CTE time

In Run mode, each CTE-based federate will perform a Time Advance Request, wait for the next CTE time and then check that a Time Advance Grant has been received, before simulating the next time step as shown in Figure 10.

Discussion: This pattern requires that the Master is also required to be connected to the CTE.

4. Spatial Design Patterns

This section describes the patterns for handling spatial information. Space simulations may include assets that operate on or about celestial bodies other than the Earth. Therefore, there is no common reference frame of convenience for all space simulations. Moreover, 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 using a common parent reference frame in order enable interaction. For example, a simulation of a ground station on the Earth sending commands to a spacecraft orbiting Mars may simulate the ground station in an Earth-centered frame and the spacecraft in a Mars-centered frame but relates these two frames using a Solar System Barycenter frame. The Space Reference FOM accomplishes this using two patterns:

1. Reference frames are explicitly specified using object instances of a ReferenceFrame object
2. Reference frames are organized using a replaceable and extendable tree of ReferenceFrame objects

4.1. Reference Frames explicitly specified using object instances

Requirement: Different models in a federation need to perform calculations related to positions that are geographically dispersed. It is conceptually and computationally inconvenient to perform all calculations using the same coordinate system.

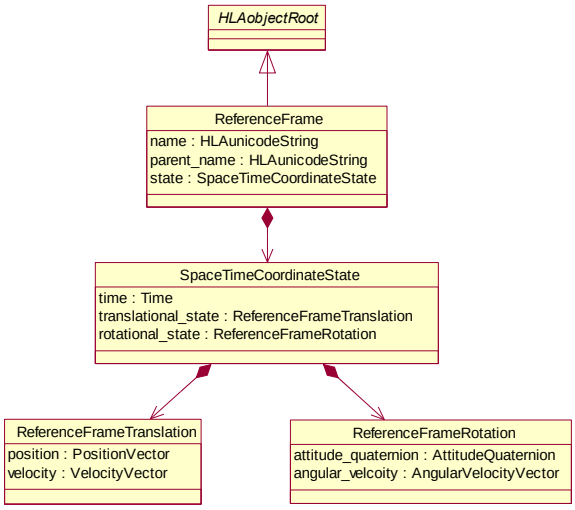

Figure 11: Reference Frame

Description: Create one object instance for each reference frame that is required. Each ReferenceFrame is identified using a name. The Space Reference FOM defines a syntax for creating unambiguous reference frame names. Each ReferenceFrame object specifies a parent ReferenceFrame by name and the ReferenceFrame's translational state (position and velocity) and rotational state (attitude and rotation rate) relative to the parent ReferenceFrame. Quaternions are used to describe orientation to avoid the singularities of Euler coordinates. ReferenceFrames also specify the Terrestrial Time (TT) congruent with the translational and rotational state.

Discussion: Many other FOMs use an implicit coordinate system, for example geodetic coordinates (latitude, longitude, and altitude). This becomes very inconvenient if you, for example, were to simulate the behavior of a rover on the surface of the moon using such Earth-based coordinates.

4.2. Replaceable and Extendable Tree of Reference Frames

Requirement: 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.

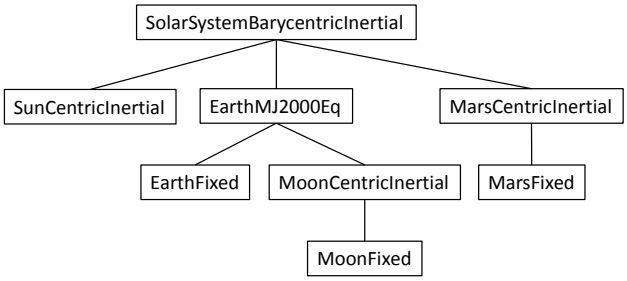

Figure 12: Tree of Coordinate Systems TBD

Description: 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. Translation between any two reference frames can be performed by traversing the graph to a common parent.

New reference frames can be dynamically added into the tree as needed. The actual tree may be different between different scenarios.

To assure that all federates agree on the relative states between reference frames, a specialized federate calculates and publishes the translational and rotational states of the reference frames in the tree. The Space reference FOM requires that a designated Root Reference Frame Publisher exists in any federation. A reference to the root reference frame is stored in the ExCO object.

Discussion: One advantage of this pattern is the opportunity to develop and reuse federates that simulate, for example, the bodies of the solar system. Alternate federates may provide different models with different fidelity. One disadvantage is the calculations needed to convert between different reference frames. However, in many space federates, this may always be required.

5. Discussion

5.1. Simpler and more advanced versions of the patterns

The initialization patterns described in this paper are available in three different versions:

1. This paper that presents these patterns "as simple as possible, but not simpler". This makes the principles easier to understand.
2. The Space Reference FOM that provides the same patterns with all details that are necessary to implement them, in particular with the HLA service calls described. Anybody that wishes to implement a federate compliant with the Space Reference FOM should study these carefully.
3. The IMSim document that presents even more extended versions, also including check pointing. This is interesting background reading for the advanced developer. Note that there are a number of differences between these patterns and the Space Reference FOM.

5.2. Comparison to defense training federations

The most widely used FOM in the defense training domain is the SISO Real-time Platform Reference FOM [9,10]. There are major differences between the Space Reference FOM and RPR FOM. Most of them are due to the fact that the RPR FOM replicates the behavior and information model of the earlier DIS [11] standard (which is based on the even older SIMNET framework) and seeks to maintain backwards compatibility. The Space Reference FOM represents a view of simulation interoperability that is at least one or two decades newer. Some key differences are:

Reliable data exchange. The information exchange in the Space Reference FOM uses reliable communication, as opposed to best effort transportation in the RPR FOM.

Causality and repeatability. The use of time managed delivery of updates and federate time advance in the Space Reference FOM guarantees correct delivery order between federates, which is required for causality and repeatability. Not only may RPR FOM updates be delivered in the past of a federates logical time, it may even be lost.

Well-managed set of federates. The required federates in a Space FOM federation are explicitly checked during startup. No corresponding mechanism is specified in the RPR FOM.

Synchronization. Federates may take some time to go between run, freeze and shutdown. The Space Reference FOM guarantees that no simulation starts before all systems are ready. In the RPR FOM, all federates can be seen as "free-running" and starting their simulation independently after a freeze. Coordinated shutdown isn't supported.

Support for soft real-time and Central Timing Equipment. The Space Reference FOM allows for any mix of soft real-time synchronized and central timing equipment. The RPR FOM is commonly used with GPS time or similar for time stamping, but there is no coordination between the GPS time and the delivery of updates with such time stamps.

Use of multiple reference frames. The Space Reference FOM supports any number of reference frames, together with a system for translation between them. This enables simulations to use reference frames that are computationally convenient for them. The RPR FOM implicitly use geocentric coordinates, which may work for Earth centric simulations, but are inconvenient for space simulation. Note that the RPR FOM supports Relative Spatial attributes for relating entities to other "parent" entities.

6. Conclusions

A number of design patterns and principles from the Space Reference FOM have been presented. The patterns relate to three areas: execution control, time management and spatial design with reference frames. All simulations in the space domain need to implement solutions for these areas, even for running standalone. When several space simulations are federated, handling of initialization, time and space are the fundamental areas that need to be addressed, before higher level processes, like space travel, can be addressed. This is why these areas are the focus of the first version of the Space Reference FOM.

6.1. Sharing knowledge inside and outside of the Space simulation community

The main purpose of the paper is to introduce the patterns and design principles to developers of distributed simulation in the space domain. The Space Reference FOM is already getting attention from developers and organizations outside of the current SISO PDG, which is promising. A prerelease of the Space Reference FOM was also used in the SEE 2017 university outreach program.

A secondary purpose is to share them with simulation developers from other domains. Through SISO and other organization we can exchange ideas, learn from each other and advance the state of the art.

References

[1] B. Möller, E. Z. Crues, D. E. Dexter, A. Garro, A. Skuratovskiy, A. Vankov. A First Look at the Upcoming SISO Space Reference FOM. Proceedings of the SISO 2016 Simulation Innovation Workshop (SIW), Orlando, Florida, USA, September 11-16, 2016.

[2] B. Möller, A. Garro, A. Falcone, E. Z. Crues, D. E. Dexter. Promoting a-priori interoperability of HLA-based Simulations in the Space domain: the SISO Space Reference FOM initiative. Proceedings of the 20th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (ACM/IEEE DS-RT), London, UK, September 21-23, 2016, ISBN: 978-150903504-5.

[3] IEEE: "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)", IEEE Std 1516-2010, IEEE Std 1516.1-2010, and IEEE Std 1516.2-2010, www.ieee.org, August 2010.

[4] Simulation Exploration Experience (SEE) project, [online], available at http://www.exploresim.com/

[5] SISO BPDSP, www.sisostds.org

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[7] SISO Space Reference FOM, Draft of May2017

[8] Daniel E. Dexter, Tony E. Varesic "Integrated Mission Simulation (IMSim). Multiphase Initialization Design with Late Joiners, Rejoiners and Federation Save & Restore.", NASA Simulation and Graphics Branch (ER7) Software, Robotics and Simulation Division (ER) Engineering Directorate, Lyndon B. Johnson Space Center, May 7, 2015

[9] SISO: "SISO-STD-001.1-2015, Standard for Real-time Platform Reference Federation Object Model (RPR FOM)", www.sisostds.org, September 2015.

[10] Björn Möller et al.: "RPR FOM 2.0: A Federation Object Model for Defense Simulations", 2014 Fall Simulation Interoperability Workshop, (paper 14F-SIW-039), Orlando, FL, 2014.

[11] IEEE: "IEEE standard for, Distributed Interactive Simulation – Application Protocols", IEEE Std 1278.1-2012, www.ieee.org, 2012

Author Biographies

BJÖRN MÖLLER is the Vice President and co-founder of Pitch Technologies. He leads the development of Pitch's products. He has more than twenty-five years of experience in high-tech R&D companies, with an international profile in areas such as modeling and simulation, artificial intelligence and web-based collaboration. Björn Möller holds a M.Sc. in Computer Science and Technology after studies at Linköping University, Sweden, and Imperial College, London. He is currently serving as the chair of the Space FOM Product Development group and the vice chair of the SISO HLA Evolved Product Development Group. He was recently the chair of the SISO RPR FOM Product Development Group.

EDWIN "ZACK" CRUES has over 25 years of professional experience in developing spacecraft simulation and simulation technologies. Zack is currently a member of the Simulation and Graphics branch at NASA's Johnson Space Center in Houston, Texas where he leads the development of simulation technologies and the application of those technologies in the simulation of NASAs current and proposed crewed spacecraft. He has developed hundreds of models and simulations for NASA spacecraft including Shuttle, International Space Station (ISS), Orion, Altair, Morpheus and the Multi-Mission Space Exploration Vehicle. Zack's recent research focus has been developing and applying distributed computation and distributed simulation technologies. This includes a large-scale distributed simulation of NASAs proposed human space exploration missions. Zack also has international experience in developing simulations of European Space Agency launch systems and Japanese Aerospace Exploration Agency spacecraft.

DAN DEXTER is an engineer in the Simulation & Graphics Branch in the Software, Robotics and Simulation Division of the Engineering Directorate at NASA's Johnson Space Center in Houston, Texas. He has over 22 years of software and simulation development experience ranging from nonlinear signal and image processing, distributed supercomputing, and flight related software to national and international distributed simulations. He is the principal developer of the TrickHLA software package, a NASA developed middleware software package for using the HLA distributed simulation standard with NASA standard M&S tools.

ALFREDO GARRO is an Associate Professor of Computer and Systems Engineering at the Department of Informatics, Modeling, Electronics and Systems Engineering (DIMES) of the University of Calabria

(Italy). He is currently Visiting Professor (from January to October 2016) at NASA Johnson Space Center (JSC), working with the Software, Robotics, and Simulation Division (ER). His main research interests include: Modeling and Simulation, Systems and Software Engineering, Reliability Engineering. His list of publications contains about 100 papers published in international journals, books and proceedings of international and national conferences. He is vice chair of the SISO Space Reference FOM Product Development Group. He is the Technical Director of the "Italian Chapter" of INCOSE.

MICHAEL MADDEN is the Chief Scientist for the Simulation Development and Analysis Branch at NASA Langley Research Center. He holds a B.S. and M.S. degree in Aerospace Engineering from Virginia Tech. Mr. Madden has 24 years of experience developing simulations of wide variety of aerospace vehicles for human-in-the-loop, hardware-in-the-loop, and distributed applications. His areas of interests include physical modeling of vehicles and their operating environments, simulation and real-time software architectures, and avionics software for both aircraft and spacecraft.

ANTON SKURATOVSKIY is a senior software engineer with RusBITech. After his 10-year service in the Air Force, he worked for D-3-Group and GTI6 companies since 1999 participating in research activities focused on using distributed simulation technologies in aerospace applications including support to ATV-ISS simulation and ground controller training. Currently at RusBITech he is working on both HLA and DDS middleware.