Promoting a-priori interoperability of HLA-based Simulations in the Space domain: the SISO Space Reference FOM initiative

Björn Möller
Pitch Technologies
Repslagaregatan 25
582 22 Linköping, Sweden
bjorn.moller@pitch.se

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

Edwin Z. Crues, Daniel E. Dexter
Simulation and Graphics Branch (ER7)
Software, Robotics, and Simulation Division (ER)
NASA Johnson Space Center
2101 NASA Road 1, Houston, TX
{edwin.z.crues, daniel.e.dexter}@nasa.gov

***Abstract*— Distributed and Real-Time Simulation plays a key-role in the Space domain being exploited for missions and systems analysis and engineering as well as for crew training and operational support. One of the most popular standards is the 1516-2010 IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA). HLA supports the implementation of distributed simulations (called Federations) in which a set of simulation entities (called Federates) can interact using a Run-Time Infrastructure (RTI). In a given Federation, a Federate can publish and/or subscribes objects and interactions on the RTI only in accordance with their structures as defined in a FOM (Federation Object Model). Currently, the Space domain is characterized by a set of incompatible FOMs that, although meet the specific needs of different organizations and projects, increases the long-term cost for interoperability. In this context, the availability of a reference FOM for the Space domain will enable the development of interoperable HLA-based simulators for related joint projects and collaborations among worldwide organizations involved in the Space domain (e.g. NASA, ESA, Roscosmos, and JAXA). The paper presents a first set of results achieved by a SISO standardization effort that aims at providing a Space Reference FOM for international collaboration on Space systems simulations.**

Keywords— Space, Interoperability, High Level Architecture, Federation Object Model.

I. INTRODUCTION

Simulation is increasingly used in the Space domain for several purposes. It is exploited for analysis and engineering, from mission level down to individual systems and subsystems, where simulation plays a key tool through the whole lifecycle, from the concept exploration phase to mission design and operation. Another example is training of flight crew and flight controllers, where simulation plays a crucial role as the Space domain is characterized by scarce training opportunities, high cost of real equipment, dangerous scenarios and emergency operations. In particular, great benefits derive from the exploitation of distributed simulation approaches as they allow for combining models from the same or different sources (within the same organization or between different organizations), to run simulation between different locations, and to promote scalability, modularization and usability [4]. Indeed, several distributed simulations have been developed for example for docking vehicles with the ISS and for mission training, in many cases with participants from several nations [1], [10], [11].

To facilitate the integration of distributed simulation models within a common architecture the IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) has been defined [6].

In the HLA standard, a distributed simulation is called Federation and it is composed of several HLA simulation entities, each called a Federate, which can interact using a Run-Time Infrastructure (RTI). The RTI represents a backbone of a Federation execution and provides a set of standard protocols and services to manage the communications and data exchange between Federates. Each Federation has a Federation Object Model (FOM) that is created in accordance with the Object Model Template (OMT) provided by the standard. A FOM defines the set of information that can be exchanged and managed in a Federation; indeed, a federate can publish and/or subscribe objects (and related attributes) and interactions (and related parameter) on the RTI only in accordance with their structures as defined in the FOM [8].

Although HLA is increasingly used in the Space domain to meet the requirements for simulation interoperability in the US, Europe and to some extent in Asia, so far different organizations and projects have developed incompatible FOMs to meet their specific needs but increasing the long-term cost for interoperability. In this context, the availability of a reference FOM for the Space domain will enable the development of interoperable HLA-based simulators and related joint projects and collaborations among worldwide organizations involved in the Space domain (e.g. NASA, ESA, and ASI).

However, there is currently no reference FOM that addresses Space exploration, since, for example, the RPR FOM

is restricted to defense operations in a geocentric environment running in real time [9].

To fill this void, a Product Development Group (PDG) has been recently activated in SISO with the aim to provide a Space Reference FOM for international collaboration on Space systems simulations [14]. Members of the PDG come from several countries and contribute experiences from projects within NASA, ESA and other organizations. Participants represent government, academia and industry.

Moreover, the PDG benefit from the wide experience gained in the "Simulation Exploration Experience" (SEE) (formally Smackdown) SISO's university outreach program [1], [3], [10]. Indeed, competencies from NASA and other organizations have been reused in the SEE project to create a core Space Reference FOM. Approximately fifteen different university teams have successfully used this FOM for six consecutive integration projects during the last six years, thus providing a solid base for the SISO standardization initiative.

The Space Reference FOM shall support interoperability for space simulations. This includes federations executing in real-time as well as federations executing in logical-time (including as-fast-as-possible). The primary focus is on training, analysis, mission support and engineering although other types of usage, like test and concept exploration may also be supported to some degree.

The standard consists of two parts: (i) the SISO Standard for the Space Reference FOM Federation Agreement. This is a natural language, human readable overview, description and specification of the FOM; (ii) The Space Reference FOM. This is a set of computer-interpretable HLA IEEE 1516- 2010 FOM modules (XML files), intended for consumption by HLA runtime infrastructure and other software tools.

These outcomes are expected to make collaboration politically, contractually and technically easier. It is also expected to make collaboration easier to manage and extend.

The Space Reference FOM provides for baseline interoperability. Project specific modules that extend it can be added as needed and commonly used extensions can be added to the standard as they mature.

The first version of the standard under release focuses on handling of time and space; in particular, the Space Reference FOM provides the following: (i) a flexible positioning system using Coordinate Reference Frames for arbitrary bodies in space, (ii) a naming conventions for well-known Reference Frames, (iii) definitions of common time scales, (iv) federation agreements for common types of time management with focus on time stepped simulation, and (v) support for physical entities, such as space vehicles and astronauts.

The rest of the paper is organized as follows. An overview of the Space Reference FOM is given in Section II. In Section III, the part of the standard for handling Coordinate Reference Frames is presented. Section IV introduces the time scales exploited in the standard to represent the time coordinate of entities in Space. Section V is devoted to discuss main issues and chosen solutions for managing the time advancement of Federation executions (including real-time simulations) compliant with the standard. The part of the standards that models physical Space entities, such as space vehicles and astronauts, is reported in Section VI. Finally, conclusions are drawn and future work delineated.

II. The Space Reference FOM: An overview

In the HLA standard, a FOM is a specification defining the information exchanged at runtime to achieve a given set of federation objectives. A FOM includes the definition of Objects (*ObjectClass*) and Interactions (*InteractionClass*) [8]. An *ObjectClass* is composed of a set of *attributes* whose values define the state of the object at any point during the simulation execution; whereas, an *InteractionClass* defines an event that a Federate can generate or react to during a simulation. It is composed of a set of parameters that define its characteristics. These two kinds of information are exchanged through a *publish/subscribe* model by using the services provided by the RTI [6]. A Federate can register an Object, which is an instance of an *ObjectClass*, and then change the values of its *attributes*. Other Federates that are subscribed to that *ObjectClass* can discover the related instances and then receive attribute value updates. The *Interactions* work in a similar way, except that interactions have associated a set of *parameters* and are not persistent (an interaction is "destroyed" after being consumed).

The Space Reference FOM defines a hierarchy of *object* and *interaction classes* for HLA that provides interoperability between simulations in the Space systems domain. It is designed to link simulations of discrete physical entities into distributed collaborative simulations of complex Space related systems. Its capabilities include representations of:

- Physical entities such as mobile surface systems, atmospheric flight systems, space flight systems, lifeforms, infrastructure elements, and interactions between them.
- Collections of individual entities collected as a single aggregate entity.
- Environmental objects and processes.
- Communications between entities.
- Emissions generated by entities.
- Logistics, including repair and resupply.

The introduced *object* and *interaction classes* are grouped in separate FOM modules (XML files) so as to allow for a more flexible and effective management of the standard proposal as well as of its extension. At the moment the following four modules have been defined: *SISO_SpaceFOM_datatypes*, *SISO_SpaceFOM_environment*, *SISO_SpaceFOM_entity*, *SISO_SpaceFOM_switches*.

The *SISO_SpaceFOM_datatypes* module provides the definitions of: (i) simple data types, for handling the main scalars physical quantities (*Angle, Mass, MassRate, MassMomentOfInertia, Length, Velocity, Acceleration, Scalar, AngularRate, AngularAcceleration, Time, Energy, Power, SignalStrength, Temperature, TemperatureRate, Force, Torque*); (ii) array data types, for handling vectors physical quantities (*Vector, Matrix, PositionVector, VelocityVector,*

AccelerationVector, *AngularVelocityVector*, *AngularAccelerationVector*, *InertiaMatrix*, *ForceVector*, *TorqueVector*); (iii) fixed record data types, for handling the *spacetime* coordinates and states of reference frames (*SpaceTimeCoordinateState*, *ReferenceFrameTranslation*, *ReferenceFrameRotation*, *AttitudeQuaternion*). Moreover, the definition of the HLA logical *timestamp* and *lookahead* time are also provided (both represented as 64 bits integers: *HLAinteger64Time*). The above introduced data types are used for *object attributes* as well as *interaction parameters* and adopt the International System of Units (SI) wherever possible.

The *SISO_SpaceFOM_environment* module provides the fundamental data types used to represent the basic physical environmental properties associated with space-based simulations. For instance, any position of an entity in a Space FOM simulation is related to a particular *reference frame*. Many different reference frames can be used in a federation execution. The Environment FOM Module specifies how these are represented (see Section III).

The *SISO_SpaceFOM_entity* module provides the definitions of vehicle related object classes. In particular, it defines the *PhysicalEntity* object, which can be a man-made vehicle or a major sub-element of a man-made vehicle, with the following attributes: *name*, *type*, *status*, *parent reference frame*, *position*, *velocity*, *acceleration*, *attitude*, *rotational velocity*, *rotational acceleration*, *mass*, *mass rate*, *center of mass*, *inertia*, *time*. The *SpaceVehicle* object class is also defined as an extension of the *PhysicalEntity* object class that also includes the *force* and *torque* attributes. Moreover, the *PhysicalInterface* object class is also specified.

The *SISO_SpaceFOM_switches* module provides configurations settings for the Federation execution by way of global Federation execution wide switches for LRC (Local Run-Time Component) and RTI behavior [6]. Indeed, the 1516-2010 HLA standard defined a set of switches that shall be set in the FOM. These switches regulate the behavior of some of the optional actions the RTI can perform on behalf of the Federate, such as automatically requesting updates of an instance *attribute* when an *object* instance is discovered or advising the Federates when certain events occur. To facilitate easy replacement of these settings, for the modular version of the HLA 1516-2010 Space FOM the switches have been confined to the SISO_*SpaceFOM_switches* FOM module. It is expected that federations might choose to update this module based on their federation agreement.

A. Building Federations using the Space Reference FOM

The Space FOM provides a starting point for building Federations for the space domain. In addition to this, it is strongly recommended that each particular federation development team produce a Federation Agreement that specifies additional design information. This may include:

- The purpose of the Federation;
- The range of scenarios to be supported;
- Participating federates;
- Additional FOM modules that are used;
- Common data or databases that are used;
- Additional services and conventions for services exchange;
- Technical configuration data, such as networking and host information;
- Test and integration procedures.

In order to develop a Space FOM federation, development teams are encouraged to use and follow the IEEE 1730-2010 Distributed Simulation Engineering and Execution Process (DSEEP) [5]. It provides a proven seven-step process, from establishing the goals and constraints for the federation to the final execution.

It is expected that many Space federations will choose to build upon and extend the Space Reference FOM. Extensions shall be created by adding more FOM modules. Developers are strongly advised not to modify the standardized FOM modules. The main principles for adding extensions are the following:

(i). *Object* and *Interaction classes* that provide specializations of a standardized class shall be defined as subclasses of these standardized classes. They shall be defined in a project specific extension module;

(ii). *Object* and *Interaction* classes that do not provide specializations of a standardized class shall be defined as subclasses of *HLAobjectRoot*. They shall be defined in a project specific extension module;

(iii). Any new data types shall be defined in a project specific extension module.

It is recommended to group extension classes and data types in extension modules based on functional areas.

In the following Sections, a more detailed description of the main introduced FOM classes and datatypes and rules for accomplishing specific distributed simulation tasks are provided.

III. HANDLING COORDINATE REFERENCE FRAMES

Reference frames are a fundamental concept for representing when and where any physical entity exists in time and space. This representation is referred to as the state of the entity. In order to represent the state of something, it is necessary to express that state with respect to some time scale and some referent coordinate system. This combination of time and coordinate system is referred as a space-time coordinate or *reference frame*.

In the Space FOM a *ReferenceFrame* object class is defined as an observational reference frame along with a companion right-handed orthogonal set of coordinate axes that are fixed in the frame. A *ReferenceFrame* is characterized by the following attributes (see Fig. 1):

- *name*, a unique name for a reference frame instance;
- *parent_name*, a string that must correspond to the name attribute of some other *ReferenceFrame* object instance in the simulation or empty for a 'root' reference frame;

- *state*, a four dimensional representation of the *space-time coordinate state* of a reference frame with respect to its parent reference frame and expressed by using a *SpaceTimeCoordinateState* fixed record data type.

If the *parent_name* is an empty string, then only the *time* dimension has meaning.

The *time* field in the *SpaceTimeCoordinateState* specifies the simulated physical time (Terrestrial Time, TT), which represents the time dimension associated with a reference frame state. The other fields in a *SpaceTimeCoordinateState* are the *translational_state* and *rotational_state*. Indeed, many applications require knowledge of the relative attitude of one frame with respect to another. This results in three (3) dimensions of position (*translational_state*), three (3) dimensions of attitude (*rotational_state*) and one (1) dimension of time. The *translational_state* field represents the reference frame's translational state with respect to its parent frame (if the frame has no parent, this attribute is meaningless) in terms of: (i) the *position* (a *PositionVector*) of the subject frame origin with respect to the referent origin with components expressed in the referent coordinate axes; (ii) the *velocity* (a *VelocityVector*) of the subject frame origin with respect to its referent origin with components expressed in the referent coordinate axes. The *rotational_state* field represents the rotational state of a reference frame with respect to a 'referent' frame in terms of: (i) an *attitude_quaternion* (an *AttitudeQuaternion*) that specifies the orientation of the subject frame with respect to the referent; (ii) the *angular_velocity* (an *AngularVelocityVector*) of the subject frame with respect to the referent with components resolved onto the subject coordinate axes.

In a given simulation scenario (e.g. a mission to Mars), each reference frame has a parent reference frame in which its position and attitude are expressed (except for a *root* reference frame). Thus, it is possible to organize the set of reference frames, useful to represent the coordinates of the involved space entities, in a *rooted tree* structure (a *rooted directed acyclic graph*), provided that there is only a *root* reference frame and the others have at least that root as *highest* common ancestor (see Fig. 2 for an example, a detailed discussion on the reported and most common reference frames for the Space domain is outside the scope of this paper).

Given a *reference frame rooted tree* structure, it is possible to transform a *space-time coordinate* expressed in a starting reference frame to those expressed in a target reference frame. The transformation is performed by following in the tree the path that goes from the source to the target reference frame through the *lowest* common ancestor and by using the information provided by the *state* of the *ReferenceFrame*s along the path. Those transformations are based on well-known formulas from quaternion algebra [7]. This capability is very important when the mission under consideration involves entities operating in different and distant regions of space (e.g. on or close to different celestial bodies in the solar system) and that can also travel among them. Indeed, for expressing the space-time *state* of each entity with the adequate precision it is required to refer to the reference frame centered in the closest point to that entity. As an example, if a spacecraft is orbiting the Moon a good choice could be to represent its coordinate in the `MoonCentricInertial` reference frame (a reference frame centered in the Moon and with fixed axes directions independent of the Moon's rotation), then when the spacecraft leaves the moon to travel to Mars the reference frame can change to `SolarSystemBarycentricInertial` (a reference frame centered in the center of mass of the solar system and with fixed axes directions); finally, when the spacecraft reaches Mars, the right choice might be `MarsCentricInertial` (a reference frame centered in Mars and with fixed axes directions). A similar situation happens when considering a simulation involving entities operating on all the above mentioned celestial bodies.

In order to be compliant with the Space Reference FOM the following rules shall be respected: (i) all reference frames used in a Federation execution shall be documented in the associated Federation Agreement; (ii) only one root reference frame shall exist within a Space FOM compliant federation execution; (iii) all reference frame parent frames shall exist as owned published object instances when the federation execution is running (e.g. advancing time). Moreover, along with the Space FOM, a recommended set of standard reference frames is provided as well as naming conventions, defined using EBNF (Extended Backus-Naur Form) notation, to correctly construct the name of any non-standard reference frame according to the Space FOM recommendations. These guidelines should enable a-priori interoperability without limiting the flexibility in the definition of Space FOM compliant Federations.

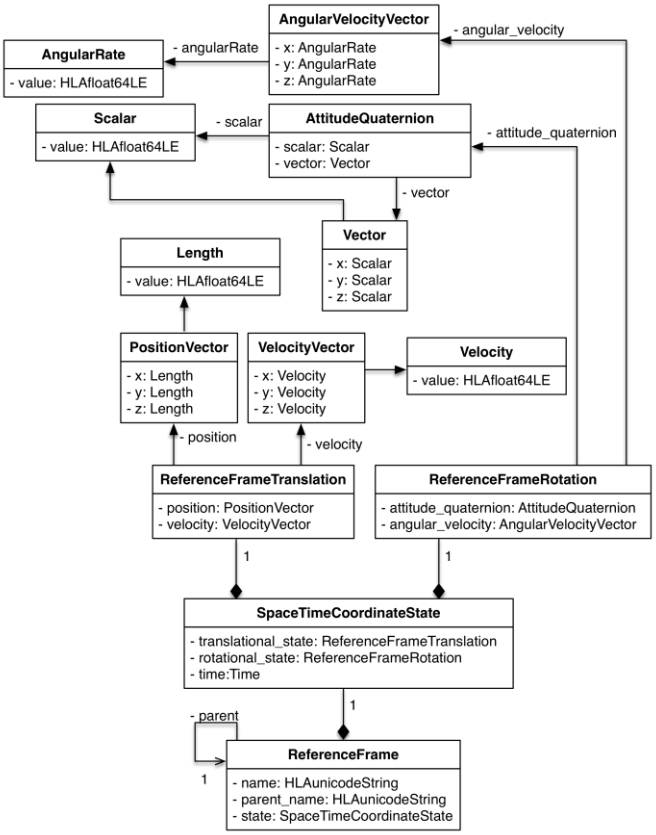

Fig. 1. Structure of a Reference Frame.

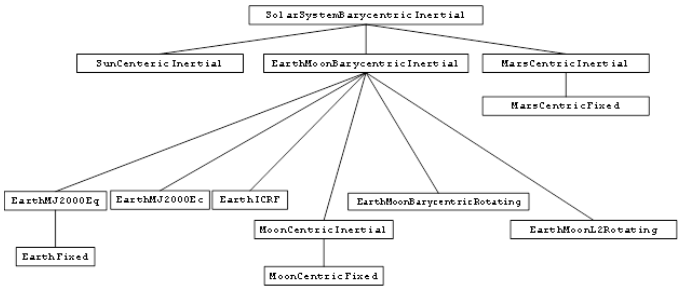

Fig. 2. A Reference Frame Tree.

IV. Time Coordinates and Time Representations

Time is a main concept to deal with in the Space domain. For a simulation, *time* is often translated into a sequentially increasing count of cycles in an execution loop. However, that is not really time, that is a count. The logical progression toward actual time would be to assign a specified quantity of time change with each execution cycle.

In a Space FOM based simulation the following different times can be distinguished:

- *Physical Time* (*PT*); it is the non-spatial dimension associated with the space-time continuum in which events are ordered in irreversible succession from the past to the present to the future. This is the fourth dimension in a space-time reference frame; the other three being the spatial dimensions representing position (see Section III). This can be conceptualized as a bidirectional infinite time line. This is the time line in which people live, work and simulate. This is sometimes referred to as "real world" time.

- *Wall Clock/Computer Time* (*CT*); unfortunately, physical time cannot be measured as an absolute value. A solution is to model the passage of time, usually with some form of oscillator, and count the oscillations with respect to some arbitrary epoch and use that as a model of time. This is often referred to as "Wall Clock" time. This is how a computer "measures" time. Since any clock, including a computer, counts from some defined starting point (epoch); a unidirectional infinite time line is the corresponding representation here.

- *Simulation Elapsed Time* (*SET*); it is the time measure associated with an individual simulation starting at zero and advancing monotonically in quantifiable steps. With cyclic executives, this is often based on some integer or floating-point counter. The counter is incremented by a predetermined amount on each cycle of the executive. This is sometimes referred to as executive time. Note that discrete event simulations usually make non-uniform steps in time. A simulation with no specified stop time would correspond to a unidirectional infinite time line. A simulation with a specified stop time would correspond to a finite time line. It is important to note that there is no substantive correlation between the passage of *Simulation Elapsed Time*, *Computer Time* or *Physical Time*. Any correlation of *Simulation Elapsed Time* to the passage time in the real world will be established in the time management policies (see Section V). *Simulation Elapsed Time* will progress at whatever rate the simulation is capable of running on a given computer.

- *Simulation Scenario Time* (*SST*); is a model within a simulation that associates the *Simulation Elapsed Time* with a representation of the problem's Physical Time. This model may provide mappings to and between multiple physical time scales (see Subsection IV.A). However, this is not the same Physical Time in the real world. This is the Physical Time in the problem space; which may be in the past, present or future. *Simulation Scenario Time* is sometimes referred to as dynamic time when used as the independent variable for numeric state propagation. This sometimes corresponds to an Ephemeris Time when used to "look-up" a priori state information for an entity in the federation execution (e.g. the position and velocity of Mars at a given time). *Simulation Scenario Time* is related to *Simulation Elapsed Time* by the starting modeled physical time, or epoch, of the simulation. Designating the simulation epoch as SST_0 results in the following relationship between *Simulation Scenario Time* and *Simulation Elapsed Time*: $SST = SST_0 + SET$.

- *HLA Logical Time* (*HLT*); is the time line used by HLA to order messages, regulate execution time advance and enable deterministic behavior in a distributed simulation. Indeed, in a time managed HLA based simulation, the RTI regulates the time advance during the Federation execution by providing to the Federates that asked to advance in time (by sending to the RTI a *Time Advance Request* (*TAR*)) a *Time Advance Grant* (*TAG*) so as to guarantee they will not receive messages with time stamps in the past [6]. HLA Logical Time often has a start epoch of 0 ($HLT_0 = 0.0$) but this is not an HLA requirement. HLA Logical time can be thought of as a Federation wide Simulation Elapsed Time. However, *HLA Logical Time* and individual Federate *Simulation Elapsed Time* will not correspond if the Federate is a late-joiner (i.e. it joined a Federation that was already advancing in time).

- *Federation Scenario Time* (*FST*); is a conceptual time associated with the physical systems being modeled in the participating Federates in the Federation execution. *Federation Scenario Time* is conceptual in that it is never computed anywhere in the federation execution but is implied by the *Simulation Scenario Time* of the individual federates and their time management schemes. This time should also be related to the HLA Logical Time by a constant offset FST_0, so as that $FST = FST_0 + HLT$. *FST* should match the *SSTs* across all federates within the federation execution to within the accuracy of the time management mechanisms (see Subsection IV.A).

A. Time Representations

The preceding Section helps to identify what time is being considered. This section discusses and defines how time is represented in a simulation time line and by the Space

Reference FOM. Just as *distance* can be measured from different starting points and with different units (e.g. meters vs. feet), *time* can be measured from different starting points (*epochs*) and with different units (seconds, angles, or days). Thus, in the Space Reference FOM, *Time* is characterized by (see Fig. 3):

- an *Epoch*, which specifies the starting point to measure time (e.g. J2000, N50, GPS, UNIX);
- a *Unit*, which specifies the unit used to measure the passage of the time from the starting epoch (Day, Hour, Minute, Second, Millisecond, etc.);
- a *Time Scale*, which is a system of assigning dates to events used to define absolute time. The Terrestrial Time (TT) time scale is used to represent absolute time for all Space FOM scenario physical time stamps; however, other time scales are also supported (e.g. TAI, UTC, UTC1, GPST, TCG) [12].

A *Time* can be a *FloatingPointTime* or an *IntegerTime* depending on the HLA datatype used to represent it and that must be interpreted on the basis of the specified *Unit* of measure.

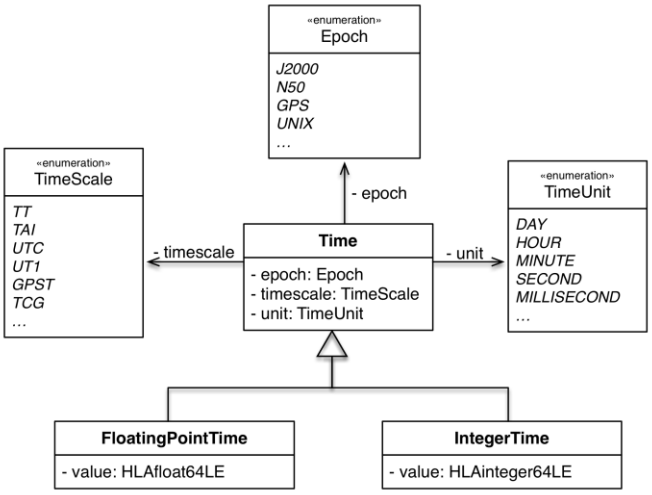

Fig. 3. Definition of a Time Coordinate.

V. Federation Time Management

Support for HLA Time Management services by Space FOM compliant federates is optional, and should be negotiated on an exercise by exercise basis. As a default and at a minimum, Space FOM federates shall operate with time stepped, clock driven, independent time advance (see [4]). Operation of the Space FOM in modes other than this time-flow mechanism is not failsafe.

Clock driven simulations are considered "real-time" because each second of elapsed execution time is equivalent to one second of time in the virtual world. Time synchronization, if it is used at all, is performed outside of the simulation itself. For example, Network Time Protocol (NTP) is often used to synchronize "wall clock" times across a Federation.

The two main time management scenario that have been considered are the following: (i) HLA Time managed real time with pacing federate (a federate that actively manages the advancement of HLA logical time during execution) and fixed time steps. This can be implemented in a *strict/conservative* way (with no frame overruns, i.e. a failure to complete processing of a HLA logical time frame during the desired real-time frame) or in an *elastic* way with catch-up on overruns (with a limited or unlimited number of allowed overruns); (ii) all federates externally synched to external reference (Central Timing Equipment - CTE) with fixed time steps. This can be implemented in *strict/conservative* way (i.e. with no frame overruns that implies *hard real time*) or in an *elastic* way with catch-up on overruns that can be in a limited (*firm real time*) or unlimited number (*soft real time*).

The first version of the standard under release focuses on a time management approach that is Time Stepped using HLA Time management and involves a *Pacing Federate* using constant time step and look-ahead. It can be locked to real time or scaled real time and elastic with catch-up on (pacing federate) overruns (i.e. soft real-time). An example is provided in Fig. 4. Also a Time Stepped approach using un-paced HLA time management with constant time step and look-ahead is supported. Future standard release will include Time Stepped using CTE and Time Stepped using CTE and HLA Time Management.

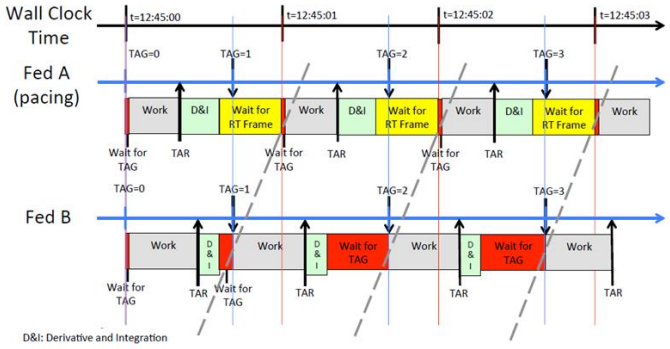

Fig. 4. HLA Time Management with early TAR from Pacing Federate.

A Space FOM compliant Federation that uses HLA Time management shall agree on a *Federation Time Step*. Given a *Federation Time Step*, the time step of each Federate participating in the Federation shall be a positive integer multiple of the *Federation Time Step:* `Federate Time Step = m * Federation Time Step` ($m \in \mathbb{N}^+$), which implies that: `Federate Time Step >= Federation Time Step`. At the same time, the *Federate Time Step* shall be a positive integer multiple of its internal *Simulation Time Step* (i.e. the *native* time step used inside the simulation when state is propagated, for example the dynamics rate): `Federate Time Step = n * Simulation Time Step` ($n \in \mathbb{N}^+$), which implies that: `Simulation Time Step <= Federate Time Step`. The inequalities introduced above guarantee a correct and effective synchronization during the Federation execution between the involved Federates. In particular, a Federate that is to be

synchronized with the Federation execution shall advance the Federate Logical Time using the HLA TAR/TAG mechanism and be HLA Time Constrained (i.e. it can receive timestamped messages) and optionally HLA Time Regulating (it can send timestamped messages). Specifically, the following rules hold to avoid dysfunctional behavior during a Space FOM based Federation execution:

- all data in the Federation that is related to the simulated scenario shall be sent by using Timestamped Order (TSO) delivery;
- all data in the Federation that is related to managing the scenario execution shall be sent by using Receive Order (RO) delivery;
- a federate that wishes to produce TSO data shall enable HLA Time Regulating;
- a federate that wishes to receive TSO data with ordering shall enable HLA Time Constrained;
- a Federate that needs to receive RO data shall Enable Asynchronous Delivery;
- a Federate that has called TAR shall not produce data before it has received a TAG;
- a federate shall only produce TSO data with time stamp greater or equal to the TAG time plus look-ahead (that specifies a lowest limit of how far in the future a federate can send messages);
- a federate shall call TAR when it has produced all TSO data in the frame to which is has been granted.

Some additional rules hold for message time stamping; in particular:

- *updates* and *interactions* shall be time stamped with the HLA Logical Time for which the data is valid and sent by using TSO delivery. The exception is represented by management interactions/attributes and initialization data, which shall be sent by using RO delivery.
- Time stamping of *updates* and *interactions* shall be done with *HLA Logical Time* (HLT) stamps that correspond to the correct Federation Scenario Time (FST). Certain data may include the federate *Simulation Scenario Time* (SST) explicitly (see Section IV), e.g. *Reference Frame* state and *Physical Entity* state (see Sections III and VI respectively).

It is worth noting that the above specified rules represent just a first set of indications for the implementation of a Federation compliant with the Space Reference FOM and that should act in a nominal way; however, a wider and refined set of rules is under definition.

VI. Representing Space Physical Entities

A *Physical Entity* is a man-made vehicle or major sub-element of a man-made vehicle. Space vehicles have two reference frames intrinsically attached to them: a 'body frame' and a 'structural frame'. Neither of these is part of the *ReferenceFrame* object hierarchy. The body frame origin is the vehicle center of mass. The structural frame is located at some well-defined point on the vehicle, but this point is not specified in the FOM. The offset of the body frame origin from the structural frame origin is captured as the vehicle's center of mass location attribute. The relative orientation of the structural frame with respect to the body frame is assumed fixed (not time varying), but it is not specified in the FOM. All dynamics of the vehicle are calculated by propagating the body frame with respect to the vehicle's 'parent reference frame' which is an object instance in the *ReferenceFrame* hierarchy and is named by the vehicle's *parent_reference_frame* attribute.

The *PhysicalEntity object class* is designed to provide a basis for the individual entities that are the principal participants in Space FOM federations. The current definition of the *PhysicalEntity object class* is based on the prototype that has been used in the SISO SEE-Smackdown project [13] and that is going to be improved and extended during the standardization activity. The core attributes shared by all entities include the entity's *position* and *attitude* with respect to a defined *parent reference frame* and a *time* tag in a defined physical *time scale* (see Section IV.A). This is sufficient to position the entity in time and space. However, *mass*, *mass rate*, *center of mass*, *inertia*, *velocity*, *acceleration*, *angular velocity* and *angular acceleration* are included to support latency compensation of the state data (a.k.a dead reckoning) – that is, to approximate its position and orientation during the period of time between state updates.

By combining position/maneuver data with classification information, the *PhysicalEntity* object class provides the set of attributes needed to visualize an entity in the virtual world. An overview of the *PhysicalEntity* attributes is provided in Fig. 5.

In the Space FOM, the *SpaceVehicle* object class extends the *PhysicalEntity* object class to provide additional attributes associated with a maneuvering spacecraft. Specifically, it provides additional *force* and *torque* attributes used to provide additional information associated with vehicle effectors and environmental effects. These can be used for both visualization and to improve state propagation between updates. Other extensions of the *PhysicalEntity* object class, as well as of the *SpaceVehicle* object class, can be defined on the basis of the specific simulation needs.

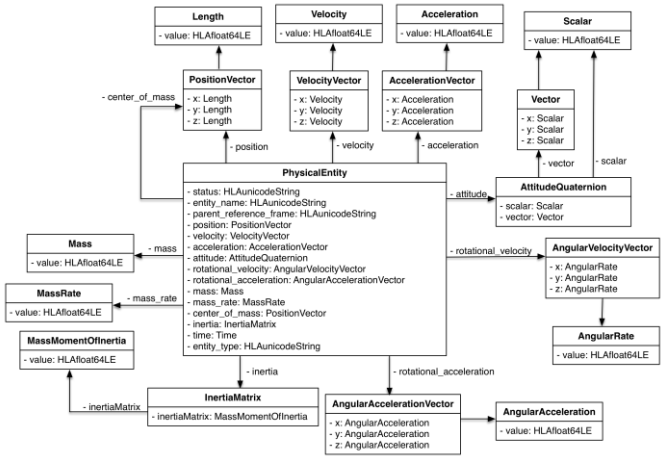

Fig. 5. Structure of a Physical Entity.

VII. Conclusion

The SISO Space Reference FOM standardization initiative presented in the paper aims at supporting the development of

interoperable simulations of complex space systems and missions, and enhance a priori interoperability among Space FOM users. The principal intended areas of use are training, analysis, mission support and engineering. However, other areas of use, like test and concept exploration, are also supported.

The benefits of the proposed Reference Space FOM include:

- Interoperability: The ability for several simulations, each focusing on particular tasks, to interoperate and jointly create a collaborative simulation with wider and richer contexts.
- Composability: The ability to build collaborative simulations from components that can be combined in different ways, with new or existing simulations, to reach a particular goal.
- Reusability: The ability to use existing simulations in new contexts. It will be possible to build generic and reusable simulations and tools for the Space domain based on the Space FOM.

The SISO Space Reference FOM initiative builds upon many years of simulation experience by professionals in government organizations, industry and academia. Early prototypes of the Space FOM have been tested in the SISO/SCS programs called "Smackdown" and "Simulation Exploration Experience". The SISO working group is going to promote and fully experiment the first release of the standard in ongoing projects involving worldwide organizations active in the Space domain (e.g. NASA, ESA, Roscosmos, and JAXA).

Finally, a software library and an HLA Development Kit that aim at easing the development of Federates and Federations compliant with the SISO Space Reference FOM are under implementation. By using these tools the developers could focus on the specific aspects and behaviors of their federates by delegating to the services provided by the underlying software layers the management of the common aspects and functionalities related to the standard.

ACKNOWLEDGMENT

The authors would like to thank all the members of the SISO Space Reference FOM (SRFOM) Product Development Group (PDG) and, in particular, Michael Madden (NASA Langley), Alexander Vankov and Anton Skuratovskiy (RusBITech).

REFERENCES

[1] L. Arguello, L. Dwedari, G. D. Lauderdale, A. Vankov, and P. Chliaev, ESA-NASA Distributed Simulation Experiment: First Results and Lessons Learned. In Proc. of the European Simulation Interoperability Workshop (EURO-SWIG), 2001.

[2] A. Falcone, A. Garro, On the integration of HLA and FMI for supporting interoperability and reusability in distributed simulation. In Proc. of the *Symposium on Theory of Modeling and Simulation - DEVS Integrative M and S Symposium, DEVS 2015, Part of the 2015 Spring Simulation Multi-Conference (SpringSim 2015)*, pp 9-16, SCS Press, 2015.

[3] A. Falcone, A. Garro, F. Longo, and F. Spadafora, Simulation Exploration Experience: A Communication System and a 3D Real Time Visualization for a Moon base simulated scenario. In *Proc. of the 18th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (ACM/IEEE DS-RT)*, pp. 113-120, IEEE Computer Society, 2014.

[4] R. M. Fujimoto, Parallel and distributed simulation systems, John Wiley & Sons, 2010.

[5] IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP). IEEE Standard 1730-2010 (2011).

[6] IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA): 1516-2010 (Framework and Rules); 1516.1-2010 (Federate Interface Specification); 1516.2-2010 (Object Model Template (OMT) Specification).

[7] J. B. Kuipers, Quaternions and rotation Sequences: a Primer with Applications to Orbits, Aerospace, and Virtual Reality. Princeton University Press, Princeton, New Jersey, 1999.

[8] B. Möller, The HLA tutorial v1.0, Pitch Technologies, Sweden.

[9] B. Möller, A. Dubois, P. Le Leydour, R. Verhage, RPR FOM 2.0: A federation object model for defense simulations. In Proc. of the Fall Simulation Interoperability Workshop (Fall SIW), pp. 233-247, 2014.

[10] R. G. Phillips, E. Z. Crues, Time management issues and approaches for real time HLA based simulations. In Proc. of the Fall Simulation Interoperability Workshop (Fall SIW), pp. 332-343, 2005.

[11] L. Rabelo, S. Sala-Diakanda, J. Pastrana, et al., Simulation Modeling of Space Missions Using the High Level Architecture. Modelling and Simulation in Engineering, vol. 2013, Article ID 967483, 12 pages, 2013. doi:10.1155/2013/967483.

[12] P. K. Seidelmann, T. Fukushima , Why new time scales? Astronomy & Astrophysics vol.265, pp. 833-838, 1992.

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

[14] SISO Space Reference FOM (SRFOM) Product Development Group (PDG) wesite, [online], available at https://www.sisostds.org/StandardsActivities/DevelopmentGroups/SRFOMPDGSpaceReferenceFederationObjectModel.aspx.