On the Execution Control of HLA Federations using the SISO Space Reference FOM

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,
NASA Johnson Space Center
2101 NASA Parkway 77058,
Houston, Texas, USA
{edwin.z.crues,
daniel.e.dexter}@nasa.gov

***Abstract*— In the Space domain the High Level Architecture (HLA) is one of the reference standard for Distributed Simulation. However, for the different organization involved in the Space domain (e.g. NASA, ESA, Roscosmos, and JAXA) and their industrial partners, it is difficult to implement HLA simulators (called Federates) able to interact and interoperate in the context of a distributed HLA simulation (called Federation). The lack of a common FOM (Federation Object Model) for the Space domain is one of the main reasons that precludes a-priori interoperability between heterogeneous federates. To fill this lack a Product Development Group (PDG) has been recently activated in the *Simulation Interoperability Standards Organization* (SISO) with the aim to provide a Space Reference FOM (SRFOM) for international collaboration on Space systems simulations. 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. The paper presents an overview of the ongoing Space Reference FOM standardization initiative by focusing on the solution provided for managing the execution of an SRFOM-based Federation.**

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

I. INTRODUCTION

The space domain is characterized by the high cost of real equipment, dangerous scenarios, scarce training opportunities, and emergency operations. As a consequence, simulation has always represented a key technology exploited for supporting space mission analysis, design and operation. In particular, due to the increasing complexity of modern space missions, which result from the cooperation of several public and private organizations of different nations, distributed simulation is becoming a fundamental asset as it allows for the combination of heterogeneous simulation models (made by the same or different organizations), running simulators from different locations, and promotion of scalability, modularization and usability (see [1], [6] for examples). In the Space domain, one of the most popular standards for implementing distributed simulations is the High Level Architecture (HLA) [3], an IEEE Standard for Modeling and Simulation (M&S). However, it is difficult to implement HLA simulators (called Federates) able to interact and interoperate in the context of a distributed HLA simulation (called Federation). The lack of a common FOM (Federation Object Model) [3] for the Space domain [4], is one of the main reasons that precludes a-priori interoperability between heterogeneous federates, developed by the different organizations involved in the Space domain (e.g. NASA, ESA, Roscosmos, and JAXA) and their industrial partners.

To fill this void, a Product Development Group (PDG) has been recently activated in the *Simulation Interoperability Standards Organization* (SISO) with the aim to provide a Space Reference FOM (SRFOM) for international collaboration on Space systems simulations [8]. Members of the PDG come from several countries, represent government, academia and industry, and contribute experiences from projects within NASA, ESA and other organizations.

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.

The first draft version of the SRFOM focuses on handling of time and space; in particular, it 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. A description of these first outcomes can be found in [4] and [5]. In addition to the previously mentioned specifications, to fully promote and support a-priori interoperability between SRFOM-based federates, the PDG is working on the definition of rules and guidelines for the execution management and control of an SRFOM-based Federation. Note that the content of this paper is based on the current SRFOM standard draft [8]. In this context, this paper, after an overview of the SRFOM initiative (Section II), discusses the proposed solutions concerning the federation executive flow (Section III), initialization (Section IV),

execution (Section V), and mode transitions (Section VI). Conclusions are drawn and future work delineated in Section VII.

II. An overview of the Space Reference FOM

The SISO Space Reference FOM (SRFOM) standard defines features to enable interoperability among HLA-based simulations (*federations*) in the Space domain [5]. This includes federations executing in real-time as well as federations executing in logical-time (including as-fast-as-possible). The main objective of the SRFOM is to support training, analysis, mission development and engineering; although other types of usage, like test and concept exploration may also be supported to some degree.

The SRFOM defines a hierarchy of *object* and *interaction classes* that are collected, according to their purposes, in separate FOM modules. This separation provides developers with a flexible and effective means for managing and extending the standard (see [4], [5]). Fig. 1 shows the architecture of the SISO Space Reference FOM along with its modules.

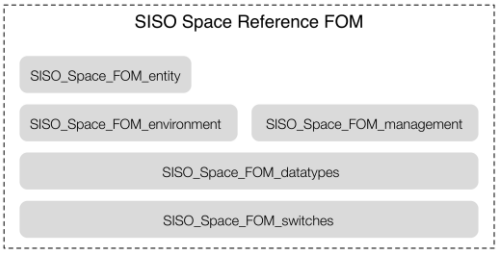

Fig. 1: Architecture of the SISO Space Reference FOM..

The *SISO_SpaceFOM_switches_module* provides configuration settings for the Federation execution by way of global Federation execution wide switches for Local Run-Time Component (LRC) and RTI behavior. The IEEE 1516-2010 standard defines a set of switches that shall be set in the FOM [3]. 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, 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.

The *SISO_SpaceFOM_datatypes* module provides the definitions of: (i) *HLA simpleDataTypes*, for handling the main scalar physical quantities, such as Angle, Mass, MassRate, Velocity and Acceleration; (ii) *HLA arrayDataTypes*, for handling vectors physical quantities, such as position, velocity and acceleration; and, (iii) *HLA fixedrecordDataTypes*, for handling the space-time coordinates and states of reference frames. Moreover, the definition of the HLA logical timestamp and lookahead time are also provided (both are represented as 64 bits integers: *HLAinteger64Time*). These 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. In particular, it defines the *ReferenceFrame HLA ObjectClass* that represents a fundamental concept for representing when and where any physical entity exists in time and space.

The *SISO_Space_FOM_management* module offers the specifications for execution control and management of *HLA ObjectClass*, *InteractionClass* and *SynchronizationPoint* instances. Specifically, it defines the base set of information necessary to coordinate federation and federate execution time lines and execution mode transitions in a SRFOM compliant federation execution.

The *SISO_SpaceFOM_entity* module provides the definitions of a space vehicle through the definition of the *PhysicalEntity* ObjectClass that represents a man-made vehicle or a major sub-element of a man-made vehicle. The current definition of the *PhysicalEntity ObjectClass* is based on the prototype that has been used in the Simulation Exploration Experience (SEE) project [2][7] and that is going to be improved and extended during the standardization activity.

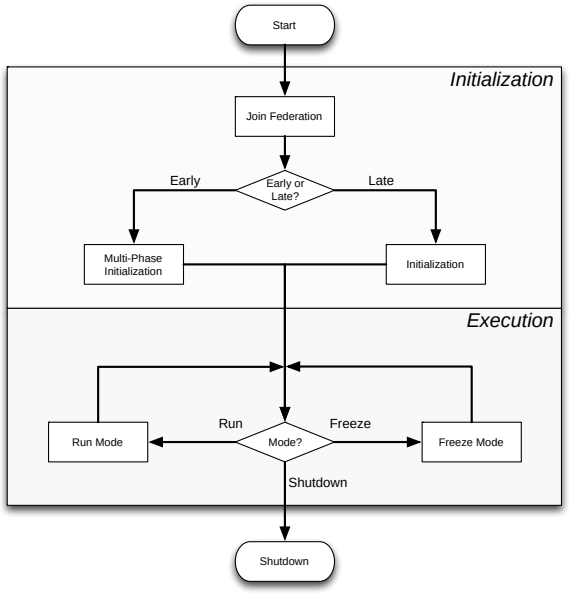

Fig. 2. Simplified Space Reference FOM Executive Flow [8]

III. Execution Control of an SRFOM-based Federation

Every simulation has an executive that controls the execution of the simulation as it starts up, goes through a defined initialization sequence, transitions into various running states, and ultimately goes through a defined shutdown sequence. At some level, a distributed simulation (e.g. a federation execution) must go through analogous processes. Interoperability between federates in a federation execution requires not only the specification of the data exchange between federates but also the specification of executive behavior. With this in mind, the Space Reference FOM defines some specifics of the execution control required for a SRFOM compliant federate.

At the highest level, the Space Reference FOM executive flow has four principal states: start, initialization, execution, and shutdown (see Fig. 1). This simplified view of the Space FOM executive architecture provides a suitable starting point for subsequent more detailed discussions.

While the Start and Shutdown states are most likely trivial entry and exit points. The Initialization and Execution states are considerably more complex. The Space Reference FOM designates the role of the Master federate as the principal federate for controlling and coordinating the federation execution. The Master federate makes use of three principal HLA mechanisms to manage execution control: (i) Execution Control Objects; (ii) Mode Transition Request Interactions; (iii) Coordination Synchronization Points.

A. *Federation Time Management*

Before discussing the execution control mechanisms used in the context of the SRFOM, it is important to recall two principal time management concepts: 1) management of Simulation Scenario Time (SST) using HLA time management mechanisms and 2) management of the federation execution with respect to the real world passage of Physical Time (PT).

Concerning the management of the Simulation Scenario Time Management (SST), the SRFOM relies on the HLA time management infrastructure for coordinating the SST progression across participating federates in a given federation execution. Federates that require time based coordination use the HLA Time Advance Request (TAR) and Time Advance Grant (TAG) mechanisms. This insures that all coordinated federates move forward with a coordinated SST. This helps to maintain consistent state information between participating time-managed federates. However, it does not regulate how fast the federation execution progresses with respect to PT. Simulations that are not paced with respect to PT will run as fast as possible (AFAP) base on the slowest executing time managed federate and the communication latencies between federates. The SRFOM supports two concepts for controlling the progression of SST with respect to PT: a Pacing Federate and Central Timing Equipment (CTE). These concepts are not mutually exclusive.

A Pacing Federate is a federate in the federation execution that employs some kind of timed wait loop to regulate the progression of SST with respect to PT. This approach is often employed through the use of the Pacing Federate's computer clock to "pace" the progression of SST with the computer's concept of the progression of PT through the use of Computer Clock Time (CCT). The Pacing Federate employs this use of CCT and the SST management mechanisms described above to "pace" the federation execution.

This method is very effective in managing the real-time progression of SST across a federation execution. However, there are limits to how well this approach can control the variation of SST progression with respect to PT across a federation execution. Specifically, the federation execution will progress in general coordination with PT but individual federates may have variations in the actual length and timing of each execution cycle. These variations are generally a result of the variability of communications latencies between federates and time management coordination implementations within the HLA Run Time Infrastructure (RTI).

In situations where more accurate control is required for real-time performance of individual federates, the SRFOM supports the use of Central Timing Equipment (CTE). There are a number of available CTE technologies (standards). However, in general, CTE provides a highly accurate coordinated timing mechanism for each computer connected through the CTE. This provides each CTE equipped computer with a commonly available and highly accurate definition of CCT. The federate's computer uses this CTE defined CCT to implement a time based wait loop to control the progression of SST with respect to PT.

B. *The Execution Configuration Object (ExCO)*

The Master federate is the principal control federate in the federation execution. The Master federate is responsible for coordinating and controlling the execution state of the federation through the use of a single instance of a published Execution Configuration Object named "ExCO" and a collection of mode transition synchronization points (see Section VI).

An ExCO, as defined in the *SISO_Space_FOM_management* module (see Section II), is a standard HLA object class which defines the base set of parameters necessary to coordinate federation and federate execution time lines and execution mode transitions in a SRFOM compliant federation execution; these attributes are:

- *root_reference_frame*: Specifies the name of the root reference frame in the federation execution's reference frame tree (see [5] for a discussion on reference frames and related data structures). This frame shall remain fixed throughout the federation execution.
- *scenario_time_epoch*: This is the beginning epoch of the federation execution expressed in Terrestrial Time (TT), using the Truncated Julian Date (TJD) origin (1968-05-24 00:00:00 UTC) as the TT epoch. This simulation scenario time (SST) epoch corresponds to HLA logical time (HLT) 0. All joining federates shall use this time to coordinate the offset between their simulation scenario times (SST), their simulation elapsed times (SET) and the HLA logical time (HLT).
- *current_execution_mode/next_execution_mode*: This is the current/next running state of the federation execution in terms of a finite set of states expressed as an ExecutionMode enumeration value.
- *next_mode_scenario_time*: This is the time for the next federation execution mode change expressed as a simulation scenario time (SST) reference. This value is only meaningful for going into freeze; exiting freeze is coordinated through a synchronization point mechanism.
- *next_mode_cte_time*: This is the time for the next federation execution mode change expressed as a Central Timing Equipment (CTE) time reference. The

standard for this reference shall be defined in the federation agreement when CTE is used.

- *least_common_lookahead*: This is used in the computation to find the next HLA logical time (HLT) boundary available to all federates in the federation execution. This is used to synchronize federates in a federation execution to be on a common logical time boundary.

C. The Mode Transition Request (MTR) Interaction

The ModeTransitionRequest (MTR) interaction is used by participating federates, that are not the Master federate, to request a federation execution mode transition. An MTR can be sent at anytime during initialization or execution but only certain MTR requests are valid at certain times (see TABLE I.). The MTR contains one parameter, the *execution_mode* that can have one of the following 3 valid values: EXEC_MODE_RUNNING, EXEC_MODE_FREEZE, EXEC_MODE_SHUTDOWN. Of these three valid mode requests, only 7 combinations of current mode and requested mode are valid (see TABLE I.).

D. Coordinating Synchronization Points

The Master federate uses a defined set of synchronization points to specify federate wide coordination points in the executive initialization and execution process flow. The Space Reference FOM specifies the following 6 execution control synchronization points; 4 are used during initialization (see Section IV) and 3 are used during execution mode transitions (See Section VI):

- *initialization_started*: Used to indicate that the initialization phase of an SRFOM compliant federation execution has been started. This synchronization point (sync-point) is not created until all federates required by the master federate have joined the federation execution. Once this occurs, the master federate announces this sync-point for all federates that have already joined the federation execution. All federates in the sync-point group must achieve this sync-point prior to proceeding with federate and federation execution initialization.
- *initialization_completed*: This synchronization point (sync-point) is registered by the federation execution Master federate after all the early joining federates have achieved the "initialization_started" sync-point. This signals to any late joining federates that they can now proceed to the current execution mode of the federation execution. This sync-point will never be achieved.
- *objects_discovered*: This synchronization point (sync-point) is used to mark the point at which all required objects have been discovered by all the federates taking part in the initialization process. This is necessary to insure the root reference frame is owned by or discovered by the federation execution Master federate prior to publishing and sending out the first update of the ExCO.
- *mtr_run/mtr_freeze/mtr_shutdown*: These are used to synchronize the mode transition to EXEC_MODE_RUNNING, EXEC_MODE_FREEZE, and EXEC_MODE_SHUTDOWN respectively. These synchronization points (sync-points) are registered by the federation execution Master federate upon receipt of a valid MTR interaction after sending out the associated ExCO update. Upon receiving the ExCO for the mode transition and at the associated transition time, all federates must achieve the related sync-point prior to going into the specified mode; an exception is represented by the transition to *shutdown* that does not involve the achievement of any synchronization point (see Section VI).

IV. The Initialization Phase

The complexity of an initialization process for a simulation can range from something as simple as setting initial simulation parameter values to the complex cyclic evaluation of initial conditions based on an iterative determination of state dependencies. The Space FOM initialization framework has to have the flexibility to support from simple to complex initialization methodologies. An overview diagram of the SRFOM initialization specification is shown in Fig. 3.

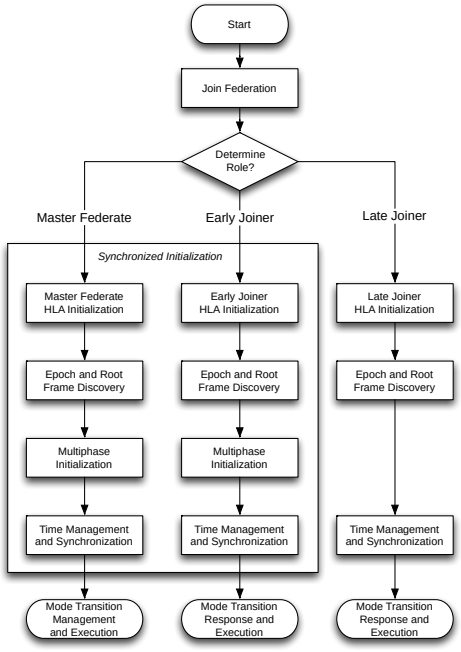

Fig. 3. Space Reference FOM Initialization Overview [8]

The SRFOM initialization process begins with the Start entry point. The next step for any federate is to join the federation execution. The SRFOM specification for joining a federation execution is a multistep process with an iterative component to avoid a potential race condition. A flow chart of the Space FOM federation join process can be seen in Fig. 4. This process starts with connecting to the HLA Run-Time Infrastructure (RTI). Strange as it may seem, the next step is to attempt to destroy the federation execution. This is done to clean up any orphaned federation executions that might exist.

The next step is to attempt to create the federation execution, ignoring a federation already exists error. The create will only succeed for the first federate. Next, join the federation execution. Note that there is a potential race condition between the creation of a federation and joining. If another federate comes in and destroys the federation execution before the current federate joins, a "Federation Execution Does Not Exist" exception will occur. If this occurs, the federate can loop back to the create step and try again. Once the federate has successfully joined the federation, the federate shall enable asynchronous delivery.

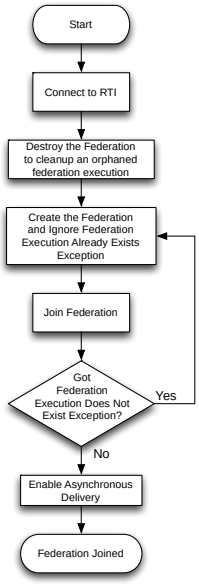

Fig. 4. Join Federation Process [8]

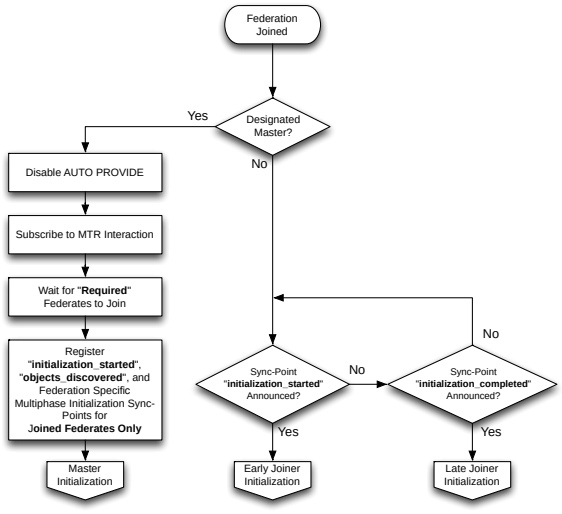

Fig. 5. Role Determination Process [8]

The "Join Federation" step is followed by the determination of the role of the participating federate. A flow chart of the role determination process can be seen in Fig. 5. This step determines if a federate is a "Master" federate (as determined by the Federation Agreement), "Early Joiner" federate, or "Late Joiner" federate. The role of Early Joiner versus Late Joiner will be determined by the timing of the federate's joining the federation execution and the federate's designation of either being, or not being, a required federate. The list of required federates shall be documented in the federation execution's Specific Federation Agreement. Any federate that plays a key regulatory role in the federation execution should be a required federate. By definition, both the Master and Pacing federates shall be required federates.

If the federate is a required federate, then it will always be an Early Joiner. If the federate is not a required federate but joins into the federation execution before the Master federate discovers the last required federate, the federate will also be an Early Joiner. All other federates will be Late Joiners.

Once a federate has determined its role, each federate will proceed to its designated initialization process: *Master*, *Early Joiner*, or *Late Joiner*.

A. The Master Federate and Early Joiners initialization

The *Master* federate has special responsibilities in the coordination and control of a Space FOM compliant federate execution initialization process. After creating and or joining the federation execution, the Master federate: (i) checks the state AUTO PROVIDE attribute in the Run Time Infrastructure (RTI), saves off the value, and then disables AUTO PROVIDE; (ii) subscribes to the Mode Transition Request (MTR) interaction. The Master federate then waits until all other required federates have joined the federation execution. All required federates and any other federates that join the federation execution prior to the last required federate will be designated as Early Joiner federates (see Fig. 5). The Master federate then begins to register and achieve specific synchronization points in a specific order to coordinate the initialization process.

Immediately after the last required federate has joined, the Master federate registers the "initialization_started" synchronization point with only the currently joined federates; these are Early Joiners. The "initialization_started" synchronization point is used to mark the start of the initialization process and the point from which all subsequent joined federates are considered "late". This eliminates a potential race condition for joining federates. Any other federates, not in the "initialization_started" synchronization point set, are designated to be Late Joiners.

The Master federate also registers an "objects_discovered" synchronization point with only the Early Joiner federates. The "object_discovered" synchronization point is used to mark the point at which all required objects have been discovered by all the federates taking part in the initialization process. To support the multiphase initialization process, the Master federate will also register any federation execution specific multiphase synchronization points with only the Early Joiner federates. These synchronization points shall be documented in the federation execution's Specific Federation Agreement.

The next step in the Space FOM initialization process is to setup and initialize the HLA infrastructure needed by the Master federate to both operate as a federate and fulfill its

responsibility as the coordinating authority for the federation execution (see Fig. 6).

The Master federate begins by waiting for the announcement of the "initialization_started", "objects_discovered", and multiphase initialization synchronization points. The federate then sets up any needed RTI handles. The next step is publishing the Execution Control object (see Section III.B).

The next step is to publish and subscribe any federate specific object classes. This will be followed by the reservation of all federate specific object instance names, waiting for the instance name reservation success or failure callbacks. At this point, the federate will register the federation specific object instances. The federate now waits for all required object instances to be discovered. Finally, the federate achieves the "objects_discovered" synchronization point and waits for synchronization. This insures that all objects required by those federates participating in early initialization have been discovered and that those federates will receive any updates for those object instances.

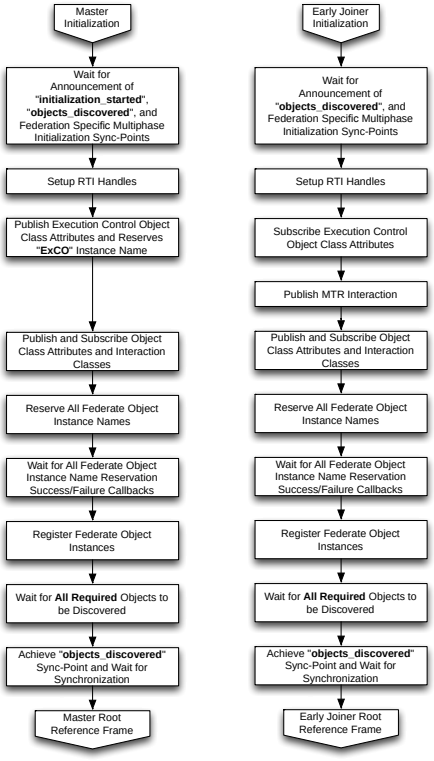

Fig. 6. Master and Early Joiner HLA Initialization [8]

In order to protect against an ExCO mode transition race condition between the Master federate publish of the ExCO and the early joiner federates subscription to the ExCO, the Master federate shall not send an ExCO update prior to achieving "objects_discovered" synchronization point.

In order to prevent a potential deadlock condition and to support shutdown at any time after the federation is synchronized on the "objects_discovered" synchronization point, all federates shall check for ExCO mode transitions in any wait loops. This includes waiting for Time Advance Grant (see Section V).

In order to prevent potentially complex mode recovery schemes during initialization, no mode transitions other than EXEC_MODE_SHUTDOWN are allowed prior to completion of the early joiner initialization process (see TABLE I.). Specifically, the Master federate shall not send an ExCO mode transition other than EXEC_MODE_SHUTDOWN prior to the registration of "initialization_complete" synchronization point.

The Master federate is now ready to establish the federation execution simulation scenario time epoch and discover the federation execution's root reference frame (see [5]). Once determined, the Master federate updates and publishes the ExCO with this information. Since the Early Joiner federates will wait on the ExCO update, the Master federate and the Early Joiner federates should be positioned to begin the multiphase initialization process that is a loop on a predetermined number of data exchanges between the Master federate and any Early Joiner federates.

Having completed multiphase initialization, the Master federate will then setup HLA time management and synchronize the federation execution in preparation for transitioning to an execution state (see Fig. 7). At this point, the Master federate will achieve and wait for the "initialization_started" synchronization point. Now, the Master federate registers the "initialization_complete" synchronization point. This is a marker synchronization point. It should never be achieved by any federate. Any Late Joiner federates that have come in during the initialization process will get the announcement of "initialization_complete" and be released to start their initialization process.

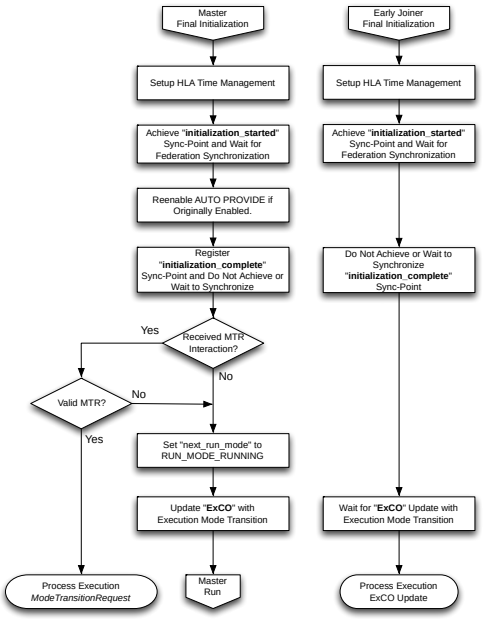

Fig. 7. HLA Time Management, Synchronization and Trans. to Execution [8]

At this point, the Master federate checks for any ModeTransitionRequests (MTRs) received during initialization. If the MTR is valid then the Master federate

proceeds according to the request (see Section VI), else if the MTR is not valid or no MTR is received during initialization, the Master federate sets the "next_execution_mode" in the ExCO to EXEC_MODE_RUNNING and updates the ExCO.

An Early Joiner federate has an initialization flow similar to that of the Master federate. Some notable exceptions are the processing of Mode Transition Request interactions and publishing ExCO updates (see Fig. 6 and Fig. 7).

This marks the end of the initialization process: the Master federate and early joiner federates will then proceed according to the related execution mode (see Section VI).

B. Late Joiner Federates

Any federate that joins into the federation execution after the Master federate recognizes the last required federate and registers the "initialization_started" synchronization point is considered a Late Joiner (see Fig. 5) and then have to wait for the announcement of the "initialization_completed" synchronization point. This is a marker synchronization point that should never be achieved by any federate. Any Late Joiner federates that have come in during the initialization process will get the announcement of the "initialization_completed" synchronization point and start their initialization process.

The Late Joiner initialization process is much like the Early Joiner initialization process but without the intermediate synchronization with the Master and there is no predefined multiphase initialization step. This makes the Late Joiner initialization process a shorter and simpler process flow, as shown in Fig. 8.

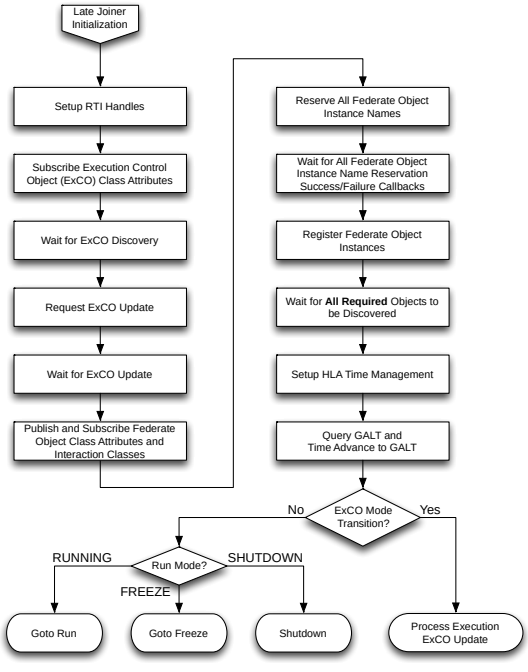

Fig. 8. Late Joiner Initialization [8]

V. THE EXECUTION PHASE

Once initialization is completed, all federates transition into one of three possible execution modes: run, freeze, or shutdown. The general execution mode flow can be seen in Fig. 9. Note that this is a general flow and that there will be differences between the execution flow between federates depending on their role in the federation execution and their capabilities. For instance, some federates might have Central Timing Equipment (CTE) while others do not.

As with most simulation architectures, the Space FOM execution architecture is composed of executive looping constructs; in this case, two executive loops or modes: a execution mode and a freeze mode. A federate exits the execution phase by making a mode transition to shutdown.

Once a federate enters execution mode, the federate enters into an execution loop that starts with waiting for a Time Advance Grant (TAG) to the last Time Advance Request (TAR) to a specific (current) federation HLA Logical Time (HLT) that will correspond to the current Simulation Scenario Time (SST). After receiving the TAG, the federate then performs any federate specific computation related to a running state. This is usually in the form of function calls (Run Jobs). Upon completion of the current SST computations, the federate makes a TAR to the HLT that corresponds to the next SST. If the federate is tied to a real-time clock, like a Pacing federate, the federate waits until the Compute Clock Time (CCT) reaches the time corresponding to the requested SST. In this case, the CCT may be either the computers clock or Central Timing Equipment (CTE). Finally, the federate checks if an ExCO update has been received with a valid mode transition during this run loop. If so, control returns to the mode selection process. If not, execution returns to waiting for the TAG to begin the next loop.

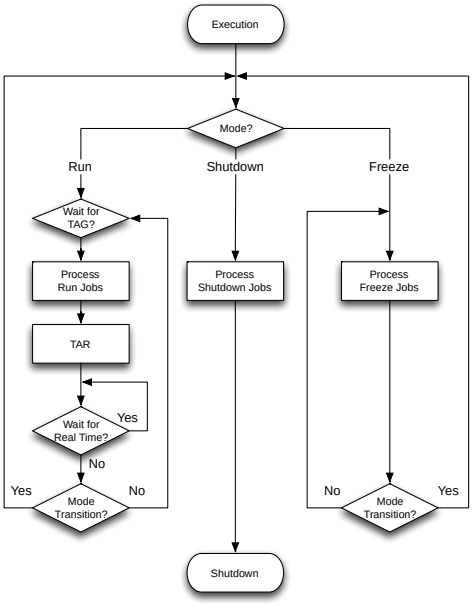

Fig. 9. Space Reference FOM Execution Overview [8]

Once a federate enters freeze mode, the federate enters into an execution loop that starts by performing any federate specific computation related to a freeze state. The federate will remain in freeze mode until an ExCO update has been received and is processed to transition the federate to a different execution mode. Note: time does not advance when in freeze mode. However, the Computer Clock Time (CCT) will advance. This means that the CCT reference point with respect to the HLT and SST time lines will have to be reset. This applies to both internal computer clocks or Central Timing Equipment (CTE) based clocks.

Once a federate enters shutdown mode, the federate performs any federate specific computation related to shutting down and then terminates. There are no mode transitions from shutdown mode.

VI. Mode Transitions

As shown in Fig. 1, an SRFOM compliant federate's executive state can be characterized by two principal executive phases: initialization and execution. However, this characterization is a little too high level for a functional implementation. Indeed, an SRFOM compliant federate will exist in one of five (5) executive initialization or execution modes: uninitialized, initializing, running, freeze or shutdown. In general, the initialization mode transitions (uninitialized and initializing) are handled internally to each federate conditioned upon the role of the federate (Master, Early Joiner, or Late Joiner) and its' progression through the initialization process. Specifically, the transition from uninitialized through initializing is gated by sync-points, ExCO updates and the progression of the federation execution through the SRFOM initialization process described in Section IV. In contrast, the execution mode transitions (running, freeze or shutdown) can be triggered through mode transition requests from any participating federate in the federation execution (see Fig. 9).

Mode transitions are controlled using two principal mechanisms: ExCO attribute updates and MTR interactions (see Section III.B and III.C respectively). Any federate can request a mode transition by issuing an MTR interaction requesting a transition to another execution mode (running, freeze or shutdown). However, the Master federate may ignore an MTR if the federation execution state is not appropriate for it (see TABLE I. on [8]). A Master federate thus is the only federate that can control mode transitions using the ExCO singleton object instance in the federation execution specifying the mode transition that will be handled by the other federates.

TABLE I. Master Mode Transition Request Validation Matrix

MTR Interaction Mode / **ExCO Current Mode**	MTR_GOTO_RUN	MTR_GOTO_FREEZE	MTR_GOTO_SHUTDOWN
EXEC_MODE_UNINITIALIZED	ignore	ignore	accept
EXEC_MODE_INITIALIZING	ignore	accept	accept
EXEC_MODE_RUNNING	ignore	accept	accept
EXEC_MODE_FREEZE	accept	ignore	accept
EXEC_MODE_SHUTDOWN	ignore	ignore	ignore

VII. Conclusion

With reference to the ongoing SISO Space Reference FOM (SRFOM) standardization initiative, this paper focuses on the main issues and proposed solution for managing and controlling an SRFOM-based federation execution. In particular, after discussing the basic mechanisms adopted by the standard for execution control (Execution Control Objects, Mode Transition Request Interactions, Coordination Synchronization Points) the executive flow of an SRFOM based federation is presented by describing in detail the initialization and execution phases along with execution mode transitions. The proposed solution for an effective federation execution control builds upon many years of simulation experience by professionals in government organizations, industry and academia. Early prototypes of these solutions have been tested in the SISO/SCS programs "Simulation Exploration Experience". The SISO working group is going to promote and fully test the first release of the standard in ongoing projects involving worldwide organizations active in the Space domain (e.g. NASA, ESA, Roscosmos, and JAXA) where the SRFOM is expected to make collaboration politically, contractually and technically easier and, at the end, to fully enable a-priori interoperability in the Space domain.

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, 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.

[3] 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).

[4] B. Möller, E. Z. Crues, D. E. Dexter, A. Garro, A. Skuratovskiy, and A. Vankov, "A First Look at the Upcoming SISO Space Reference FOM," In Proc. of the SISO 2016 Simulation Innovation Workshop (SIW) (IEEE/ACM DS-RT), Orlando, Florida, USA, September 11-16, 2016.

[5] B. Möller, A. Garro, A. Falcone, E. Z. Crues, and D. E. Dexter, "Promoting a-priori interoperability of HLA-based Simulations in the Space domain: the SISO Space Reference FOM initiative," In Proc. of the 20th International Symposium on Distributed Simulation and Real Time Applications (IEEE/ACM DS-RT), pp. 100-107, IEEE Computer Society, 2016.

[6] 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.

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

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