Extending the NATO FOM for Distributed Synthetic Training to support reinitialization of unit tasking

Karl Söderbäck, Oscar Bergman, Björn Löfstrand

Pitch Technologies
Repslagaregatan 25
58222 Linköping
Sweden

karl.soderback@pitchtechnologies.com, oscar.bergman@pitchtechnologies.com,
bjorn.lofstrand@pitchtechnologies.com

Keywords: HLA, NATO NETN, RPR-FOM, FOM Modules, Simulation, Federate, Federation, NATO FOM, Distributed Synthetic Training, DST, Initialization

ABSTRACT: *The NATO Allied Modelling and Simulation Publication AMSP-04 Ed B (NETN FOM) defines a set of HLA FOM modules that extend and complement the SISO RPR-FOM. A key module is Entity Tasking and Reporting (NETN-ETR), which represents tasking instructions sent to simulated entities while complying with the basic HLA rule of not directing interactions to a specific federate. In NETN, the federate responsible for modeling a simulated entity will receive the tasking instruction and manage its execution. In the NETN FOM, these tasking instructions are sent using HLA interactions. However, the state of planned tasks, currently executing tasks, and completed tasks are not published in the federation and are maintained only within the federate responsible for the unit. In recent exercises using NETN-ETR, the need for reinitializing federates with the existing planned and current tasks have been identified. This paper describes a proposed update of the NETN-FOM and associated design patterns and federation agreements required to support a centralized backup and reinitialization of task states.*

1 Introduction

1.1 Background

Efficient and effective use of Modelling & Simulation (M&S) capabilities requires policies, common services, shared data, and standards for interoperability and reuse.

When designing a distributed simulation environment, three main questions need to be considered:

- HOW is simulation data exchanged?
- WHAT simulation data is exchanged?
- WHEN is the data exchanged?

The HOW relies on standards related to networking and services for coordinated transfer and control of data exchange and is agnostic with respect to WHAT type of data is shared. The WHEN are patterns of simulation interplay [1] captured in federation agreements and design specifications to meet requirements for a specific simulation environment.

SISO and the NATO Modelling and Simulation Group (NMSG) have a Technical Cooperation Agreement (TCA) related to standards development. The NATO AMSP-01 "NATO M&S Standards Profile (NMSSP)" [2], provides a list of recommended M&S-related standards, including the following:

- NATO STANAG 4603 / IEEE 1516 High-Level Architecture [3] **(HOW)**
- STANREC 4800 / AMSP-04 [4] includes an HLA Federation Object Model – the NATO FOM **(WHAT)**
- AMSP-04 also includes modules describing various patterns of interplay as the foundation for federation-specific agreements **(WHEN)**

AMSP-04 "NATO Education and Training Network Federation Architecture and FOM Design" (NETN FAFD) [4] specifies a set of HLA FOM Modules providing standard interfaces for representing simulated entities, events, and other real-world objects, processes, and phenomena models. It also provides standard interfaces and patterns for interplay between systems in a federated distributed simulation. AMSP-04 is a modular reference federation agreement that contains a set of HLA FOM Modules that extend and complements the SISO-STD-001 RPR-FOM v2.0 [5].

AMSP-04 Ed B includes the following modules:

Module	Description
NETN-BASE	Common definitions of datatypes and extensions of the RPR-BASE FOM Module.
NETN-Physical	Representation of Physical Entities in a federated distributed simulation.
NETN-MRM	Aggregate level entity simulation, aggregation, and disaggregation of units. Division and merging of unit resources.
NETN-COM	Representation of Communication Networks and the status of communication links.
NETN-METOC	Representation of weather conditions and primary effects of weather on terrain, on water surfaces, in the atmosphere and subsurface water conditions.
NETN-CBRN	Representation of CBRN release, detection, effects, and protective measures in a federated distributed simulation.
NETN-LOG	Negotiation, delivery, and acceptance of logistics services between federates modelling different entities involved in the service transaction.
NETN-TMR	Negotiated and coordinated transfer of attribute modelling responsibility between federates.
NETN-SE	Representation of persistent abstract geographical objects that can be (re-)used and referenced for specifying locations, paths, etc. The module also includes the representation of facilities with a function or capability to perform activities.
NETN-ETR	Interface for sending simulation tasks to entities represented in a federated distributed simulation.
NETN-ORG	Representation of the state of units including command structure and relationship between organizations.
NETN-AIS	Represent vessel traffic in a simulation using AIS messages.

Maintenance and updates of AMSP-04 are coordinated by the NATO Modelling and Simulation Coordination Office (MSCO), managed by the NATO Modelling and Simulation Group (NMSG), and performed as NATO Science and

Technology Organization (STO) technical activities in support of the NMSG Modelling and Simulation Standards Subgroup (MS3). In addition, AMSP-04 is currently maintained on GitHub by NATO Research Task Group MSG-191, and all release and development versions are publicly available.

1.2 Principles of HLA

High-Level Architecture (HLA) [6] is a standard developed specifically for building distributed federated simulations. These simulations are called Federations and require the following components:

- A Run-time infrastructure (RTI), a software that provides standard HLA services to manage common distributed simulation tasks, e.g., synchronized data exchange.
- Federates are individual simulation systems applications and services that use the HLA services to interoperate.
- A Federation Object Model (FOM), a single or a set of modules that specifies the data to exchange in the federation.

The FOM itself contains two major types of concepts:

- An object class is a collection of information (attributes) that represents the state of an entity that persists over time. E.g., a car can be represented by specifying a position and fuel level. These attributes can be updated and published via the RTI to all the other federates subscribing to updates for that object class.
- An interaction class represents events; individual interactions are shared with federates using HLA services. An interaction can contain data in a way similar to an object class, but the interaction is not related to a persistent object in the federation.

1.3 Entity Tasking in the NATO FOM

Entity tasking is defined in the NETN-ETR FOM Module and is part of the AMSP-04 NETN FOM. The ETR module defines a set of HLA interaction classes and provides a standard interface for sending simulation instructions as tasks to entities in an HLA federation.

The main benefit of using NETN-ETR is the ability to separate applications that provide tasking from the actual simulation of the task by a federate currently having the modelling responsibility of the entity. Sending tasking information to the federation instead to a specific application allows the separation of simulation user interfaces from the simulation application and potentially allows multiple simulation systems to be controlled from a single common simulation GUI.

NETN-ETR was originally developed by FFI, Norway and TNO, The Netherlands and has successfully been used to support exercises such as Viking.

ETR defines common low-level tasks that can easily be interpreted and executed by simulators that model the behaviour of entities. It also defines a set of reports to provide status information, including the status of the tasks being executed by simulated entities. An executing simulated entity can be either a physical entity, e.g., platform or lifeform, or an aggregate organizational unit, such as a Battalion.

The simulated entities are represented as HLA object class instances. The NATO FOM extends the SISO RPR-FOM v2 [5] representation of Aggregate Entities and Platforms and includes a unique identifier (UUID). The NETN-ETR FOM module uses the unique identifier to direct tasking interactions to specific entities in the simulation.

The interactions are organized into four categories, see figure 1.

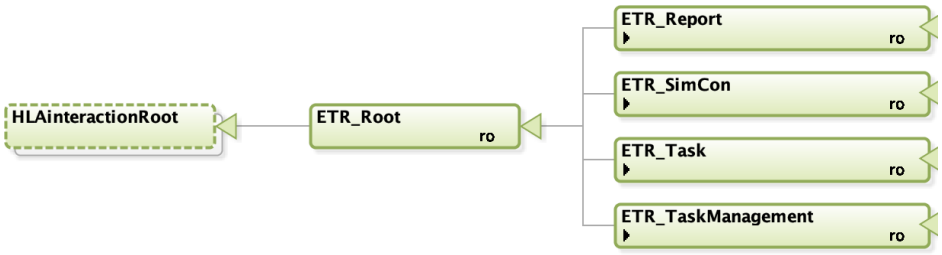


Figure 1. NETN-ETR base interactions.

These categories can be described as follows:

Category	Description
ETR_Report	Interactions related to reporting tasks.
ETR_SimCon	Interactions related to simulation control, for example, magic move.
ETR_Task	Interactions related to specific tasks, for example, MoveToLocation.
ETR_TaskManagement	Interactions related to the management of tasks.

This paper will only focus on the interactions in the ETR_Task category, but the reader is encouraged to study all aspects of the ETR module to gain a full understanding of the way these interactions are used [7]. Figure 2 shows the pattern for sending an ETR_Task interaction in a federated simulation.

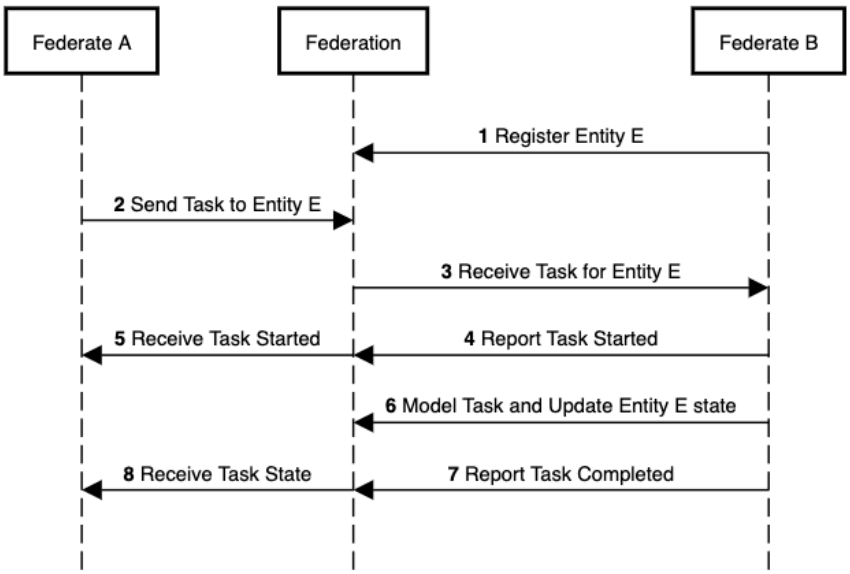


Figure 2. Pattern for sending ETR tasks in a federated simulation.

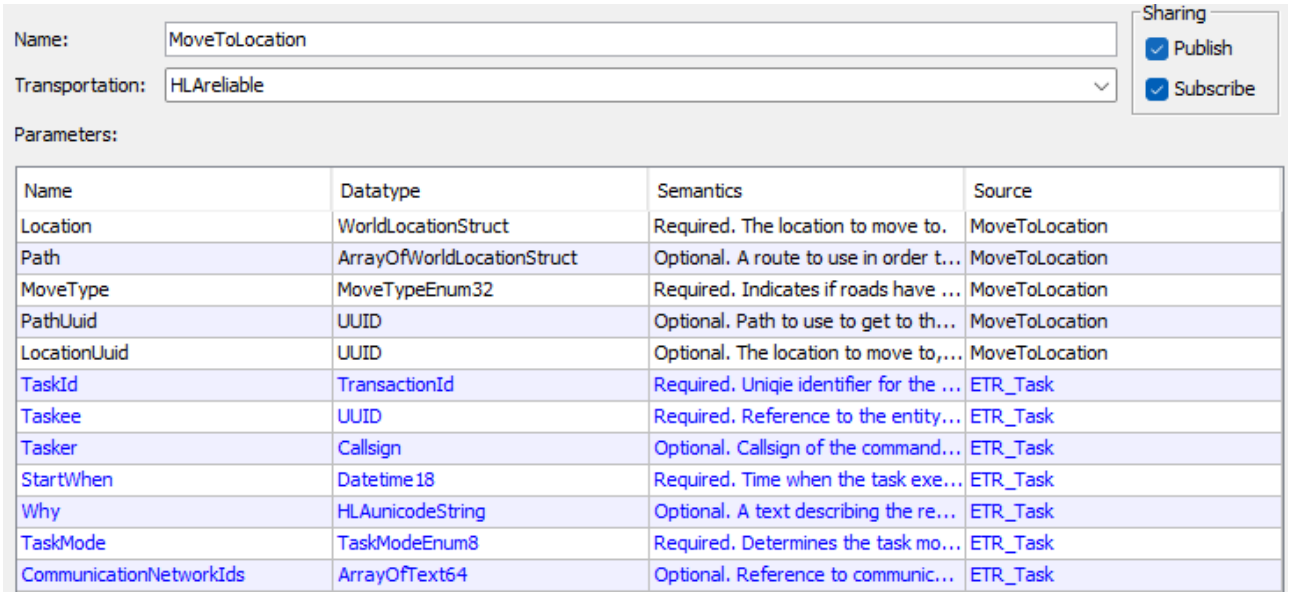
The pattern for entity tasking requires a unit to be registered before it can receive tasks; in Figure 2, this is represented in step 1 by Federate B registering Entity E.

When this entity is registered, Federate A sends (step 2) a task to the newly registered entity via the Federation to Federate B (step 3).

The task data contains required parameters as specified in the ETR_Task interaction class and task-specific additional data defined in each Task subclass in the ETR FOM module. The required parameters are:

- *TaskId* – A unique identifier for this task.
- *Taskee* – A unique identifier for the unit intended to execute this task.
- *StartWhen* –Timestamp for when this task should start executing.
- *TaskMode* – Defines if the task is expected to execute concurrently with other tasks.

E.g., a MoveToLocation task would include additional data about route and speed. Figure 3 shows all parameters included in a MoveToLocation interaction as displayed in Pitch Visual OMT [8]. The blue colour indicates parameters inherited from the ETR_Task interaction class.


Name: MoveToLocation
Transportation: HLAreliable
Sharing
Publish
Subscribe
Parameters:

Name	Datatype	Semantics	Source
Location	WorldLocationStruct	Required. The location to move to.	MoveToLocation
Path	ArrayOfWorldLocationStruct	Optional. A route to use in order t...	MoveToLocation
MoveType	MoveTypeEnum32	Required. Indicates if roads have ...	MoveToLocation
PathUuid	UUID	Optional. Path to use to get to th...	MoveToLocation
LocationUuid	UUID	Optional. The location to move to,...	MoveToLocation
TaskId	TransactionId	Required. Uniqie identifier for the ...	ETR_Task
Taskee	UUID	Required. Reference to the entity...	ETR_Task
Tasker	Callsign	Optional. Callsign of the command...	ETR_Task
StartWhen	Datetime18	Required. Time when the task exe...	ETR_Task
Why	HLAunicodeString	Optional. A text describing the re...	ETR_Task
TaskMode	TaskModeEnum8	Required. Determines the task mo...	ETR_Task
CommunicationNetworkIds	ArrayOfText64	Optional. Reference to communic...	ETR_Task

Figure 3. MoveToLocation task.

A task sent to a unit is handled by the federate responsible for simulating the unit. The federate models all task-related simulation calculations and updates the new entity state in the federation.

E.g., a MoveToLocation task will produce position and status updates of the unit when moving along its route. Federates also sends reports for all tasks it models to indicate execution status (steps 4 and 5). The status reports can reflect when the task has been accepted, started, and completed.

2 Issues with Task State

In a federated system, services and applications are loosely coupled and executed in a distributed environment. Therefore, there is always a need to design the federated system to gracefully handle situations where a subset of components requires, e.g., restart or late join. For the user, these situations may cause major interruptions in the system's operation, and therefore there is a need to quickly and seamlessly handle the reinitialization of the simulation state. This includes ongoing and planned activities of simulated units.

The problem with the current NATO FOM approach is that the task state is not explicitly represented in the federation or the initialization data. Initial task information is transferred using HLA interactions (steps 2 and 3 in figure 2), and the information about task progress and the order of tasks is only stored internally in the federate performing the task. There is a need to standardize the approach for centrally managing the initialization and reinitialization of the task state in the federation.

Reinitializing task state is primarily needed for tasks that execute over a period.

E.g., A MoveToLocation task state includes not only where the unit is located but also its next waypoint in its route, i.e., how far the unit has traveled along its route. Other tasks can require more specific state information when reinitialized.

If a federate requires a restart due to a crash or for any other reason, it is important that the complete state of the simulated units can be restored. This may include information regarding previous, current, and future planned tasks. Figure 4 illustrates how the execution of a task is interrupted by a Federate Lost event.

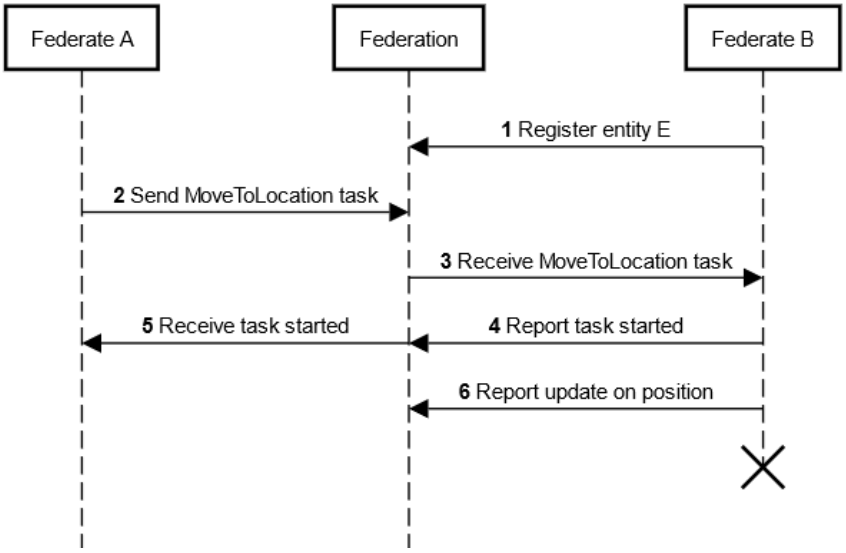


Figure 4. Federate lost.

In Figure 4, the task execution is ongoing, and restoring the federate to the same state as when it was lost is vital.

Currently, the only solution is manually recreating all unit-related tasks in a restarted federate. For example, imagine a federate representing 1000 units, all of which are performing a MoveToLocation task. If the federate requires a restart, all 1000 MoveToLocation tasks will be lost and must be recreated.

3 Solution

To achieve a persistent task state, the following aspects were considered:

- How to represent a task state in the federation.
- How to store and restore task state.
- Establishing a pattern for the (re-)initialization of federates.

A reference implementation was developed based on the use of NATO FOM with the following extensions:

- An extension of the NETN-MRM FOM module to represent the task state of entities in the federation.
- An extension to the NETN-ETR FOM module to support new data types used to represent task data in the federation.

3.1 Representing Task State in the Federation

To represent the state of tasks in the federation, both the definition of the task and the execution progress of the task is required. Two new datatypes are introduced:

- *TaskDefinition* – is a fixed record with fields for both common and task-specific data that defines the task.
- *TaskProgress* – is a fixed record containing a description of the execution progress of a task.

3.1.1 TaskDefinition

The *TaskDefinition* datatype is designed to capture all relevant parameters provided in the ETR_Task interaction. In addition, the following fields are included:

- *Activity* – reflects which activity is related to the task, based on the *AggregateMissionEnum16* datatype.
- ***Status*** – reflects the status of the task based on the *TaskStatusEnum32* datatype.
- ***DefinitionData*** – a variant record that contains all task-specific information, e.g., the route for a MoveToLocation task. The *DefinitionData* variant record uses a task-type enumerator as its discriminant. Based on the discriminant, different task-specific definition data can be represented.
- *MainTaskId* – a unique identifier of the main task in a chain of tasks.

- *PreviousTaskId* – a unique identifier of the previous task in a chain of tasks.
- *NextTaskId* – a unique identifier of the next task in a chain of tasks.

3.1.2 TaskProgress

The *TaskProgress* datatype is designed to capture the progress of ongoing tasks. It contains all the data necessary to restore ongoing task executions to their last known state. *TaskProgress* contains the following fields:

- ***TaskId*** – a unique identifier referencing the task
- ***LastUpdated*** – a timestamp representing the latest progress update
- *ExecutingFederateId* – unique identifier referencing the federate responsible for executing the task
- *ETC* – estimated time of task completion
- ***ProgressData*** – variant record containing all task-specific progress data. *ProgressData* uses the task type enumerator as a discriminant. Based on the discriminant, different task-specific progress data can be represented.

3.1.3 Task State in the Unit representation

The state of tasks is divided into planned, current, and previous tasks. Together these constitute the entire task list for an entity and reflect the entire task state.

In the proposed NATO FOM extensions, the task list is represented as attributes of the NETN_Aggregate_EXT object class as depicted in figure 4. Similar extensions are planned but not yet implemented for the NETN extensions of the RPR-FOM Platform object classes.

NETN_Aggregate_EXT			
CurrentTaskProgresses	ArrayOfTaskProgress	ps	ro
CurrentTaskDefinitions	ArrayOfTaskDefinitions	ps	ro
PreviousTasks	ArrayOfTaskDefinitions	ps	ro
PlannedTasks	ArrayOfTaskDefinitions	ps	ro

Figure 4. NETN_Aggregate_EXT, the object class with task list attributes extended from NETN_Aggregate.

The task list state is represented as four arrays:

- *CurrentTaskProgresses – TaskProgress* records for all ongoing tasks.
- *CurrentTaskDefinitions – TaskDefinition* records for all ongoing tasks.
- *PreviousTasks – TaskDefinition* records for completed and cancelled tasks.
- *PlannedTasks – TaskDefinition* records for planned tasks.

As progress is only relevant when a task is ongoing, there is no representation of progress for previous and planned tasks.

3.2 Storing and Restoring Federation State

By representing the task list state as attributes of simulated entities, this information can be saved by recording entity state updates.

The complete published federation state is stored by incrementally saving all objects and attributes as they are created, deleted, and updated during the simulation. This can be achieved by a tool such as Pitch Recorder [9].

To restore the federation state, the stored data can be re-published as represented at a specified point in time. The initialization pattern, see section 3.3, is used to share the restored state in the federation. Participating federates are then able to resume the simulation using the restored state.

3.3 Pattern for (Re-)Initializing Federates

The pattern for (re-) initialization of federates uses HLA services for Object and Ownership Management.

Initial or stored federation state data is published by an initialization federate. This includes the task list attribute data.

The suggested pattern requires one federate responsible for initialization and one or several federates responsible for simulating certain sets of entities. Figure 5 shows a simple representation of the initialization procedure.

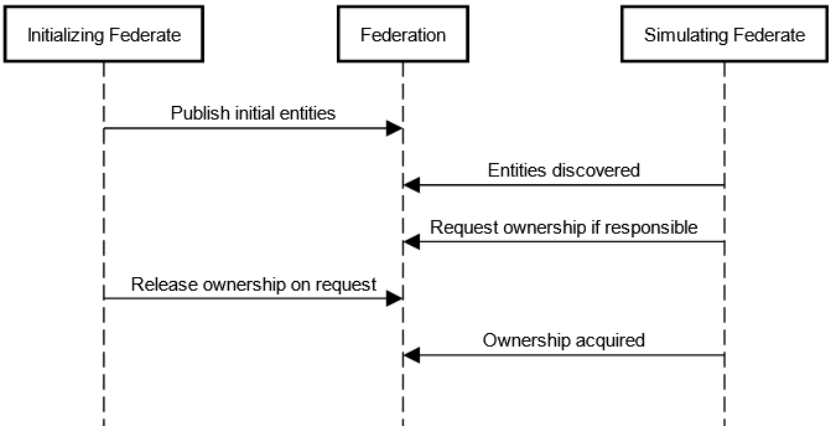


Figure 5. Ownership-based initialization procedure.

The initializing federate publishes all entities initially, with initial attributes for each entity. The simulating federates then detect when entities they are responsible for have been published, and request ownership of their attributes. The initializing federate releases ownership upon request. Once the simulating federate has acquired ownership, it can start simulating the behaviour of the entity. The initial attribute values published by the initializing federate will be used.

The FederateApplication object class in the NETN ORG module is used to describe the responsibility of simulation in the federation. It describes each simulating federate with a unique id, name, and arrays of unique identifiers for the entities said federate is responsible to simulate. By using this object class, the responsibility of each simulating federate is published in the federation.

4 Summary and Conclusions

Previously, task state persistence could not be handled through any standardized method. It was not possible to represent in a federation. This prevented storing and restoring the task state as part of the federation state from a central component. Instead, each federate with task data had to handle the task state internally.

The NATO FOM was extended to represent the task state as part of the published federations state. Additionally, a method for storing and restoring the federation state from a centralized component was proposed. Finally, a pattern for (re-) initialization of the federation state was designed. These three aspects combined provided a way to achieve a persistent task state through a standardized approach.

4.1 Conclusions

By representing the task state in the federation, it is published together with the other object attributes that are of interest in the context of storing and restoring. This provides a great advantage when managing this service in a centralized component as all data can be found and published in one place. Being able to provide centralized control also removes the need for all federates to reliably be able to store and restore their own state. Instead, all federates that comply with the pattern for (re-) initialization can benefit from the dedicated federate restoring the federation state. As the pattern is based on existing HLA services and information available in the standardized NATO FOM, the threshold to adapt is low while the utilization provides high value.

4.2 Future Work

The NATO FOM extensions made should be developed further to be integrated into future official versions of the FOM to represent the task state in the common standard. This will drive the development of future federates to utilize the newly suggested data types and attributes. Further, the method for storing and restoring the federation state as well as the (re-) initializing pattern should be shared for future federates to adapt.

5 References

[1] SISO, “SISO-STD-003-2006: Standard for Base Object Model (BOM) Template Specification (8 May 06),” SISO, 2006.

[2] NATO, “AMSP-01 (D): NATO Modelling and Simulation Standards Profile AMSP-01 Edition D version 1,” NSO, 2018.

[3] NATO, “STANAG 4603: Modelling and Simulation Architecture Standards for Technical Interoperability: HLA,” NSO, 2015.

[4] NATO, “AMSP-04 (B): NATO Education and Training Network Federation Architecture and Federation Object Model Design (NETN FAFD), AMSP-04 Edition B,” NSO, 2021.

[5] SISO, “SISO-STD-001-2015: Standard for Guidance, Rationale, and Interoperability Modalities (GRIM) for the Real-time Platform Reference Federation Object Model (RPR FOM), Version 2.0,” SISO, 2015.

[6] SISO, “IEEE 1516-2010: IEEE Standard for High Level Architecture for Modeling and Simulation — Framework and Rules,” IEEE, 2010.

[7] NATO, "NETN-ETR documentation," 4 September 2020. [Online]. Available: https://github.com/AMSP-04/NETN-ETR/blob/master/NETN-ETR.md.

[8] Pitch Technologies, "Visual OMT," [Online]. Available: https://pitchtechnologies.com/visual-omt/.

[9] Pitch Technologies, [Online]. Available: https://pitchtechnologies.com/recorder/.

6 Author Biographies

KARL SÖDERBÄCK is a Software Architect and Developer at Pitch Technologies. His work includes designing and developing High-Level Architecture (HLA) applications for distributed simulations, a recent example of such is adaptations of simulation systems to the NATO FOM for supporting the VIKING 2022 exercise. He assisted with the on-site integration and exercise support of the HLA infrastructure for VIKING exercise. Mr. Söderbäck has an M.Sc. in Computer Science from the University of Linköping (Sweden).

OSCAR BERGMAN is a Software Developer at Pitch Technologies. His work includes development of High-Level Architecture (HLA) applications for distributed simulations and development of Federation Object models (FOM). Mr. Bergman is a member of NMSG group MGS-191, his work there includes the development of the NETN-FOM standard. Mr. Bergman has an M.Sc. in Computer Science from the University of Linköping (Sweden).

BJÖRN LÖFSTRAND is the Vice President of Pitch Technologies and a senior systems architect with over 25 years of experience in modelling and simulation research, standards development, simulation architectures and design of distributed simulation in support of training, analysis, and experimentation. His work includes engaging in NATO and international M&S standards development activities to evolve and develop standards that allow the efficient design of simulation systems. These standards include High-Level Architecture (HLA), Federation Development and Execution Process (FEDEP), Distributed Simulation Engineering and Execution Process (DSEEP), Base Object Models (BOMs), Real-Time Platform Reference FOM (RPR-FOM), and several NATO M&S standards related to Distributed Synthetic Training (DST) M&S environments. Mr. Löfstrand is an NMSG Member at Large. He has published and presented over 30 papers and articles at conferences and workshops worldwide, such as ITEC, I/ITSEC, CAX Forum, and the NMSG Symposium. Mr. Löfstrand has an M.Sc. in Computer Science from the University of Linköping (Sweden).