Use Cases for the HLA Evolved Modular FOMs

Björn Möller
Björn Löfstrand
Pitch Technologies
Nygatan 35
SE-582 19 Linköping, Sweden
+46 13 13 45 45

bjorn.moller@pitch.se
bjorn.lofstrand@pitch.se

Keywords:
Interoperability, HLA Evolved, OMT, FOM, Modular FOM, RTI, GIG

ABSTRACT: The HLA Federation Object Model (FOM) describes the information that is to be exchanged during the execution of a federation. In earlier versions of HLA a federation had one monolithic FOM. HLA Evolved (the next version of HLA) introduces Modular FOMs which allow a federation to load FOM data as modules and to extend the FOM during the execution.

The modular approach to FOMs introduces many new opportunities and use cases:

Modular development of reference FOMs. Reference FOMs are very important for the reuse of federates. One example is the multitude of COTS and project specific RPR FOM based federates. They enable commercial and government users of M&S to quickly compose federations to meet their needs with low risk and at a modest cost. At the same time the sheer size of such FOMs, that may cover many aspects of the simulation, makes the development slow and costly. Modular FOMs can make this process quicker, more flexible and less costly.

Efficient handling of specialized extensions of reference FOMs. Reference FOMs are frequently modified and extended to support the simulation of local or specialized entities or services. This has resulted in a large number of uncorrelated versions of reference FOMs. As M&S users need to interoperate between sites, services and nations the handling of these non-standard versions of reference FOMs introduces risk and cost. Modular FOMs can separate out particular concerns thus preserving the reusability through reference FOMs and keeping the extension process well-controlled.

Reusable federation agreements modules. For particular aspects of a federation, it will now be possible to provide a FOM module and a corresponding federation agreement. This may be a smaller aspect, like start and stop signaling or time pacing, or larger aspects like voice communications or weather services. In this way modular FOMs can support reuse of components of simulations systems across and between federations.

Support for long-running, GIG-style federations. Today's HLA federations require the entire FOM to be loaded upon startup. In many larger systems, new capabilities need to be added over time without shutting down the entire federation. Modular FOMs allow new capabilities to be loaded when federates supporting these new capabilities join at a later point in time.

1. Introduction

The Federation Object Model (FOM) is a key part of the HLA standard, both for the older DMSO HLA 1.3 version [1] and the newer IEEE 1516-2000 [2] version. A new version of the HLA IEEE 1516 standard, with the working name "HLA Evolved" [3] is currently under development.

One of the major improvements in HLA Evolved is that the monolithic Federation Object Model (FOM) format of earlier HLA version has now been replaced with Modular FOMs, i.e. the ability to split the FOM into several parts, called FOM Modules.

A description of how FOMs relate to Federation Development and Execution Process (FEDEP) [4] and how the FOM format has evolved in earlier versions of HLA has been provided in the earlier paper "An Overview of the HLA Evolved Modular FOMs" [5]. The purpose of this paper is to summarize the concept of Modular FOMs and to provide additional insights in a number of expected use cases.

2. Overview of the FOM

The FOM describes the data that is to be exchanged during a federation execution. The FOM is one of the most important parts of the federation agreement, which is the agreement between participants in an HLA federation.

A closely related concept is the Simulation Object Model (SOM) that describes what information a particular simulation is capable of sharing (publishing and subscribing to) in a potential federation. The SOM may be used in early project steps to judge the suitability of a potential simulation for a certain purpose.

The exact format of a FOM and a SOM is described in the HLA Object Model Template which is one of the three documents that constitutes the HLA standard.

2.1 Contents of a FOM

A FOM contains the following:

- **Identification**. This is a general description of the FOM, like domain, purpose, developer, etc.
- **Object classes and attributes**. Shared object classes are described, for example Airplane. Subclasses can be introduced, for example a Jet plane that has additional attributes.
- **Interactions and parameters**. These are instantaneous events that are shared between federates. Just like Object classes they are structured into a hierarchy.
- **DDM information**. This is a description of data to be used for value based filtering, for example nationality, latitude and longitude.
- **Data types** for attributes, parameters etc. The HLA 1516-2000 standard and later supports full and unambiguous specification of data types down to the bit level, including complex data types, cardinality and padding rules. A number of predefined data types are also provided.
- **Transportation types** that can be used for attribute updates and interactions. This table contains the predefined data types HLAreliable and HLAbestEffort but can also be extended.
- **Update rates**. In HLA Evolved updates for the same object instance attribute can be provided with different update rates to different federates. This table specifies these update rates.
- **Synchronization points**. This table describes global synchronization points that can be used for example for synchronizing start up or scenario execution.
- **User defined tags**. For certain RTI services, like updates, application specific meta-data can be provided. The user defined tags table provides the data types for this data.
- **Time representation**. This table provides the data types for the simulated, logical time, as well as for time intervals.
- **Switches**. This table provides a number of standardized runtime switches that controls the behavior of the RTI.
- **Notes**. These are human-readable comments that may refer to one or more classes, attributes, data types, etc.

2.2 What's not in a FOM?

A mistake that is sometimes made about the FOM is the class versus instance description. The FOM contains information about object classes, both scenario classes (aircraft, ground-vehicles, etc) and meta-classes (federate and federation). However, it does *not* contain any information about how and when these will be instantiated.

3. Modular FOMs in HLA Evolved

HLA Evolved introduces three important module concepts:

FOM Modules. These are modules that contain FOM information and that follow the OMT DIF Schema. Note that not all tables are required to be present. When FOM Modules are loaded into a federation they shall, when combined, form a valid FOM. A MOM Module is also required in a FOM.

MOM Module. This is a module that contains the MOM information. It also contains some additional, standardized information such as predefined data types, transportation types, dimensions, etc. There is a standard MOM module that may be provided automatically but it is also possible to provide extended MOM modules.

SOM Modules. In much the same way as FOMs can be built by assembling FOM Modules, SOMs can be built from SOM modules.

A simple example of how FOM modules can be combined is shown here.

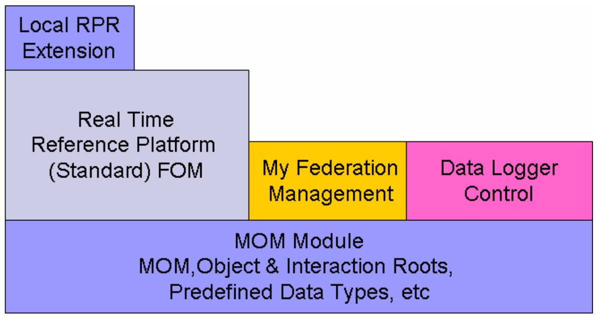

Local RPR Extension
Real Time Reference Platform (Standard) FOM
My Federation Management
Data Logger Control
MOM Module
MOM,Object & Interaction Roots,
Predefined Data Types, etc

Figure 1: FOM Module example

A more detailed example of how classes within FOM modules can build upon each other is shown here:

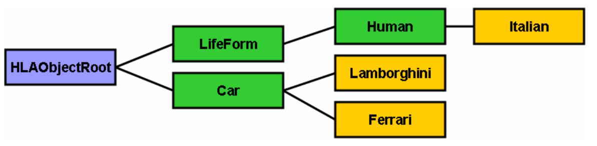


Figure 2: Classes from FOM Modules

The blue HLAobjectRoot comes from the MOM Module. The green entity classes come from one FOM Module. The Italian concepts, here shown in yellow, come from another FOM module.

3.1 Combining FOM Modules

Several FOM Modules can be combined into a new FOM and existing, monolithic FOMs can be decomposed into FOM modules. The following principles for combining the elements of FOM modules are used, based on the different tables as defined in the OMT standard:

Principle A. Exact equivalency between tables. If the table is present in more than one FOM module then the tables shall be equivalent. Principle A applies to Switches, Time representation and User Supplied Tags.

Example: One FOM module specifies that a 64-bit integer time representation shall be used. Another FOM module specifies that a 64-bit float time representation shall be used. It will not be possible to combine these FOM modules. The underlying reason is that there will be no actual agreement between federates about what logical time representation that shall be used.

If, on the other hand, the other FOM module does not provide a time representation, this means that logical time is not a concern for federates that only support that FOM module. This may, for example, be federates that monitor the state of the federation and don’t participate in any time managed operations.

Principle B. Produce the union of table elements, duplicates shall be equivalent. If the table is present in more than one FOM module the union of the elements (rows) of the table is produced. If any two elements have the same identifier (name) their content (rows) shall be equivalent. Principle B applies to Dimensions, Data types, Transportation types, Update rates, Synchronization points and Notes.

Example: One FOM module specifies an integer datatype “FuelInt” for fuel level and a fixed record data type “PositionRec” with latitude, longitude and altitude for position. Another FOM module specifies a float data type “SpeedFloat” for speed and the exact same “PositionRec” as in the first FOM. When these FOM modules are combined the resulting FOM will contain all three data types.

If, on the other hand, the second FOM module contains a different definition of the “PositionRec”, for example using geocentric coordinates, it will not be possible to successfully combine these FOM modules. The federation builders will need to go back and agree on how to represent positions or, possibly, provide different representations for different purposes.

Principle C. Produce the union of hierarchical elements, duplicates shall be equivalent or scaffolding (“empty placeholders”). Similar to principle B but this is used for elements in a hierarchy, namely object classes and interaction classes. The hierarchies of different modules are added. Consider the following object class hierarchies:

FOM Module A:

```
HLAobjectRoot
   Vehicle (Name, Speed)
```

FOM Module B:

```
HLAobjectRoot
   Vehicle – scaffolding definition
     AirCraft (Altitude, Airline)
```

Resulting FOM

```
HLAobjectRoot (PrivilegeToDeleteObject)
   Vehicle (Name, Speed)
     AirCraft (Altitude, Airline)
```

Figure 3: Combining Object Classes

If a subclass, for example Aircraft, is going to be added to the superclass Vehicle that is originally defined in FOM module A, it is possible to either *repeat* the full vehicle definition or to provide a *scaffolding* (“empty placeholder”) definition containing just the name. Note that if a class definition is repeated it must contain the exact same set of attributes or parameters. A FOM module cannot add more attributes to an already defined class. This may instead be achieved by creating a subclass. Principle C applies to object classes with attributes as well as to interaction classes with parameters.

Principle D. Table not used. This is a special case for metadata about the FOM module, namely the identification table. There is no way to algorithmically determine the purpose, version or point of contact for a FOM based on the contributing FOM modules. [1]

The complete set of rules for combining FOM Modules is given in the HLA Evolved OMT standard, not in the interface specification. One reason for this is that it is possible to combine FOM Modules without an RTI, for example in an OMT development tool.

3.2 Load operations and module life cycle

At runtime there are two ways to specify that a FOM module shall be loaded:

Create Federation Execution takes a list of FOM modules as an argument. It also takes an optional MOM module argument. If no MOM module is specified the RTI will provide the standard MOM module. This means that, in practice, most federation developers need never worry about providing the MOM in their FOMs any more.

Join Federation Execution also takes a list of FOM Modules. All modules that weren't already loaded will now be loaded. A possible design pattern is to have federates load all FOM Modules that they depend upon when joining. The FDD data loaded by one federate becomes available to all federates in the federation.

Also note that the above load operations are atomic in the sense that they either succeed with loading all of the specified FOM modules or they fail.

Note that the load operation only really cares about the FDD, not the entire FOM. This means that it shall be possible to successfully combine the FDD portion of the FOM modules. There are no requirements for example for the Notes or Semantics to be equivalent for an attribute that is defined in two different modules. If the FDD portion of the FOM modules cannot be successfully combined due to a conflict between the modules then the whole load operation will fail.

A FOM module will remain available in the federation execution until the Destroy Federation Execution service is successfully executed. The MOM module will of course be available for the entire life span of the Federation Execution.

3.3 Reporting the Current FOM/FDD

The Current FOM is the sum of the currently loaded FOM Modules. When all of the FOM modules that were part of the federation design have been loaded the Current FOM will equal the FOM of the federation.

The RTI is only responsible for combining the FDD part of the FOM Modules. The RTI also provides services for reporting the Current FDD as well as contributing FOM modules. The following MOM functionality is available for this:

For the **HLAfederate** Class:

HLAFOMmoduleDesignatorList is an attribute whose value is the list of identifiers of all FOM modules that were loaded by this federate in *the Join Federation Execution* service call.

For the **HLAfederation** Class:

HLAFOMmoduleDesignatorList is an attribute whose value is the entire list of unique identifiers of all FOM modules that have been loaded into the Federation Execution, either by *Join Federation Execution* service calls or *Create Federation Execution* calls.

HLAMOMmoduleDesignator is an attribute whose value is the identifier of the MOM Module specified in the *Create Federation Execution* service invocation or implicitly used by the RTI.

HLAcurrentFDD is an attribute whose value is the current FDD as an XML Unicode string. This string shall comply to the OMT DIF Schema.

There are also a set of MOM interactions that make it possible to request the actual contents of each FOM and MOM modules.

4. Building FOMs from FOM Modules

There is one additional FOM Module property that needs to be understood before using them. A FOM Module can be *Standalone*, which means that it can be used on its own, without any other FOM module (but a MOM module will always be required). FOMs as we know them in HLA 1.3 and HLA 1516-2000, when converted to HLA Evolved, will become Standalone FOM Modules since they can be used without other FOM Modules. They may of course now be extended by other FOM Modules.

The opposite type of FOM Module is referred to as *Dependent*. A Dependent FOM module contains a reference to a definition in another FOM Module. The OMT format allows for the following types of references:

- **Object Classes**. An Object Class can reference a superclass.

[1] It has later been suggested that a number of references to contributing FOM modules shall be included in the resulting FOM.

- **Interaction Classes**. An Interaction Class can reference a superclass.
- **Data Types**. Attributes, parameters, time representations, and user defined tags as well as data types themselves can reference data types.
- **Transportation types**. Attributes and interactions can reference data types.
- **Dimensions**. Attributes and interactions can reference dimensions.
- **Notes**. Anything in a FOM can contain a reference to a note.

Note that a Standalone FOM Module may always contain references to the MOM Module.

4.1 Suggested Block Notation

The following block notation is suggested:

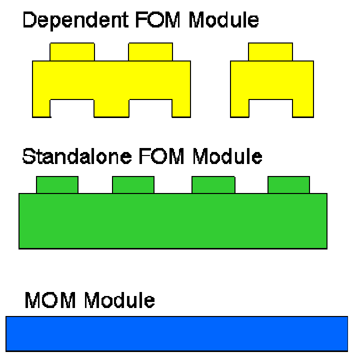
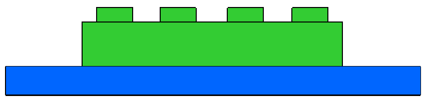


Figure 4: FOM Building Blocks

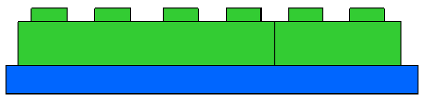
The MOM Module forms a foundation that Standalone FOM Modules can build upon. Dependent FOM Modules may in turn build upon Standalone FOM Modules.

4.2 Allowed Combinations

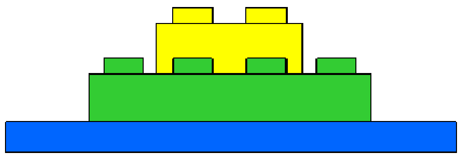
The following pictures illustrate allowed combinations:



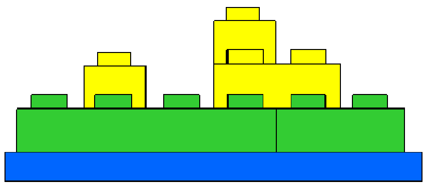
A MOM Module is always required. A Standalone module and a MOM module can be used to build a FOM.



Several Standalone FOM Modules and a MOM Module are also allowed. Remember that no conflicting definitions of concepts may take place.



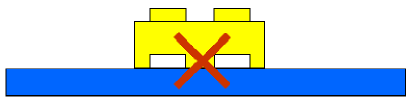
Dependent FOM Modules may be used on top of a standalone FOM Module.



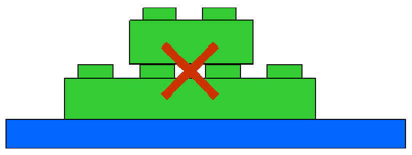
Several dependent FOMs may build upon a standalone FOM module. A dependent FOM module may build upon several standalone or dependent FOMs. A dependent FOM module may build upon a dependent FOM (but there has to be a standalone FOM at the bottom).

4.3 Disallowed combinations

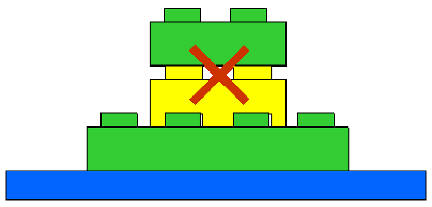
The following combinations are not allowed



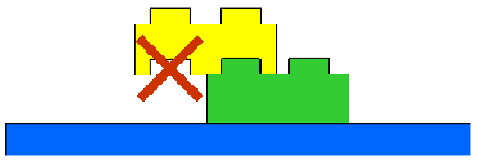
A Dependent FOM Module may not be used without a Standalone FOM module.



A Standalone FOM Module may not build upon another Standalone FOM module.



A Standalone FOM Module may not build upon a Dependent FOM module.



Every FOM concept that is referenced in a Dependent FOM Module must be provided by another (Dependent or Standalone) FOM Module.

The principles above apply to SOM modules as well with the exception that a MOM module is optional for a SOM.

5. Use Cases for Reference FOMs

5.1 Modular development of reference FOMs.

Reference FOMs are very important for the reuse of federates. One example is the multitude of commercial and custom RPR FOM [6] based federates. They enable commercial and government users of M&S to quickly compose federations to meet their needs with low risk and at a modest cost. At the same time the sheer size of such FOMs, that may cover many aspects of the simulation, makes the development slow and costly. Modular FOMs can make this process quicker, more flexible and less costly.

It will also be easier to reuse just parts of a reference FOM and to replace other parts with extended or refined FOM modules.

5.2 Managing extensions to reference FOMs

Reference FOMs are frequently modified and extended to support the simulation of local or specialized entities or services. This has resulted in a large number of uncorrelated versions of reference FOMs. As M&S users need to interoperate between sites, services and nations, the handling of these non-standard versions of reference FOMs introduces risk and cost. Modular FOMs can separate out particular concerns thus preserving the reusability through reference FOMs and keeping the extension process well-controlled.

6. Use Cases for Partial Federation Agreements

For particular aspects of a federation, it will now be possible to provide a FOM module and a corresponding federation agreement. This may be a smaller aspect, like start and stop signaling or time pacing or larger aspects like voice communications or weather services. In this way modular FOMs can support reuse of components of simulations systems across and between federations. Note that in some cases a BOM [7] may be an alternative, especially if a fuller description of the sequence of events or states in the model is needed.

It is also possible to create standardized sets of data representations that can be shared by a larger set of FOMs and federations within and between organizations.

7. Use Cases for Long-running Federations

Today's HLA federations require the entire FOM to be loaded upon startup. In many larger systems, new capabilities need to be added over time without shutting down the entire federation. Modular FOMs allows new capabilities to be loaded when federates supporting these new capabilities join at a later point in time.

8. Conclusions

A standardized format for Modular FOMs together with RTI Services and Rules has been developed as part of comment round three of HLA Evolved. This provides a new level of flexibility and many new opportunities.

The main advantage is the increased flexibility during runtime and, not the least, during the development process. Federation developers can build new modules that extend reference FOMs without modifying them. Reference FOMs can be developed by several smaller communities in different domains or for different local or national extensions. Temporary additions will not result in a multitude of similar FOMs.

Several use cases for modular FOMs have been presented. First of all the development of Reference FOMs can be greatly enhanced if different types aspects of the FOM can be handled by different exprts. Secondly any specialized extensions to reference FOMs can now be separated out, preserving the reusability of reference FOMs. Other use cases are reusable federation agreement modules, for example capturing scenario or execution handling. Finally, it will now be possible to support federations that run for extended periods of time and that still need to extend the functionality and shared object model, for example in virtual arenas.

References

[1] "High Level Architecture Version 1.3", DMSO, www.dmso.mil, April 1998

[2] "IEEE 1516, High Level Architecture (HLA)", www.ieee.org, March 2001.

[3] Roy Scrudder et. al. "Evolving the High Level Architecture for Modeling and Simulation". Proceedings of the 2005 Interservice/Industry Training, Simulation & Education Conference, Paper No. 2157, National Training Systems Association, December 2005.

[4] "Federation Development and Execution Process (FEDEP)", IEEE 1516.3, www.ieee.org, 2003

[5] Björn Möller et al. "An Overview of the HLA Evolved Modular FOMs", Proceedings of 2007 Spring Simulation Interoperability Workshop, 07S-SIW-108, Simulation Interoperability Standards Organization, March 2007

[6] SISO: "Real-time Platform Reference Federation Object Model (RPR-FOM)", Version 2.0d17, based on SISO-STD-001.1-1999, www.sisostds.org, September 2003

[7] SISO: "Base Object Model (BOM) Template Specification", SISO-STD-003-2006, www.sisostds.org, May 2006

Author Biographies

BJÖRN MÖLLER is the Vice President and co-founder of Pitch, the leading supplier of tools for HLA 1516 and HLA 1.3. He leads the strategic development of Pitch HLA products. He serves on several HLA standards and working groups and has a wide international contact network in simulation interoperability. He has twenty years of experience in high-tech R&D companies with an international profile in areas such as modeling and simulation, artificial intelligence and Web based collaboration. Björn Möller holds an MSc in Computer Science and Technology after studies at Linköping University, Sweden and Imperial College, London.

BJÖRN LÖFSTRAND is Manager of Modeling and Simulation Services at Pitch Technologies. He holds an M.Sc. in Computer Science from Linköping Institute of Technology and has been working with HLA federation development and tool support since 1996. Recent work includes developing design patterns for HLA based simulation in the future Swedish Networked Based Defense.