Processes and Tools for Management and Reuse of FOM Modules

Björn Möller
Fredrik Antelius
Martin Johansson
Björn Löfstrand
Åsa Wihlborg

Pitch Technologies
Repslagaregatan 25
S-58222 Linköping
Sweden

bjorn.moller@pitch.se
fredrik.antelius@pitch.se
martin.johansson@pitch.se
bjorn.lofstrand@pitch.se
asa.wihlborg@pitch.se

Keywords:

HLA Evolved, OMT, FOM Modules, FEDEP, Reuse, Tools

ABSTRACT: *One of the most important new features of HLA Evolved is FOM Modules. The FOM is used to describe the information that is exchanged in a federation. By making the FOM modular it is possible to focus on certain aspects of the data exchange. This makes it possible to identify, develop and isolate more general or more specific aspects of the data exchange. Examples of more general and thus more reusable modules are commonly used data types or federation management interactions. Less reusable FOM modules are for example project-specific platform extensions to the RPR-FOM.*

The first part of the paper covers processes for the development and maintenance of reusable FOM modules. These can be developed and reused in a "top-down" manner within domains (i.e. defense, space, industry, etc), groups of organizations (e.g. SISO, NATO), organizations (i.e. companies, defense components) and individual projects. Important success factors here are a well-defined lifecycle process, proper management support and a use case and test driven development approach.

It is also possible to develop reusable FOM modules from a more technical "bottom-up" perspective. Useful components like FOM elements are sometimes reused within and across federations, often even beyond the planned life span. The main reusability criteria here is "the survival of the fittest".

The second part of the paper describes how a specialized tool can be used to develop and maintain a project consisting of several FOM modules for use in a particular federation.

These modules need to be inspected, understood and verified both against the HLA Evolved standard and against each other. Since FOM modules can build upon each other it is important that a tool can help the user maintain compatibility and avoid undesirable dependencies between modules.

When managing FOM modules it is important to understand what role each FOM module plays from a reuse perspective. Is it a highly standardized module or a temporary project development? This affects which modules that should be adjusted when consistency and compatibility issues are discovered. It affects several aspects of refactoring across modules. Last but not least it affects how a tool can provide "best practices" assistance to a new user. A comparison is also made between maintaining FOM modules using general-purpose XML tools versus a specialized FOM module tool.

Finally some thoughts on the above processes and tools are given, based on the on-going work with the NATO Snow Leopard federation and other practical applications.

1. Introduction

Whenever you connect two or more systems to exchange data there will be an information exchange data model. This model may be explicit or implicit but it will always be there. For typical business systems this model may be focused on providing a simple service with a well-known set of outcomes, for example verifying a credit card transaction. For simulation systems that are required to interoperate, the information exchange may include a large number of entities and interactions, possibly describing an entire battlefield.

A large number of simulators may need to consume information from each other. The resulting effect of a small state change may, from case to case, be minimal or massive and it may sometimes be hard to predict for the original information producer. For the simple interaction between business systems a two-part point-to-point data exchange may be sufficient. For simulation systems with many participating systems on the other hand a common data bus is the optimal solution.

1.1 Challenges with hard-coded domain models

When exchanging data the most obvious approach is to create a network protocol. This specifies where in each exchanged data block each domain property should be stored, for example DIS [1]. Certain bytes in the exchanged packets may describe the marking or the position of an aircraft, in effect hard-coding the protocol to a particular solution approach for a certain domain. This makes it convenient to adapt each new simulator to the protocol since both the format and the content of the protocol are well-known in advance. The problem here is that there will always be variations in the requirements and that requirements will grow over time. For slight variations a few non-standard packets can be introduced. For applications with different requirements the protocol may not be useful at all. It is also difficult to introduce more advanced simulation services since each simulator may need to correctly implement them.

1.2 Separating out domain information

General purpose protocols, like TCP/IP [2], FTP [3], HTTP [4] and SMTP [5] have generally been very successful. Today they form the basis of the Internet and any corporate network. These typically standardize the technical level of communication. The actual domain data, like the layout and text of a web site are described on a higher level of communication. By separating the lower level protocol from the domain data each user in various application domains is given the power to describe his domain information.

The trend today is to capture the information exchange data model on a higher and more flexible level than the network packet level. This can be seen in the FOMs of the HLA standard [6], the LROMs of the TENA framework [7], the WSDL approach of Web Services [8], the Topic of the DDS [9] architecture and more. A parallel can also be drawn on the Variable Message Format (VMF) protocol of the Link-16 [10] family.

2. The HLA FOM

The information exchange model of HLA is called the Federation Object Model (FOM). It is based on the Object Model Template, which is one part of the HLA standard. The FOM describes both the information that is exchanged at runtime as well as the usage and parameters of a number of additional HLA services. It is important to understand that the FOM is only one part of the "Federation Agreement". The Federation Agreement is the document where you find descriptions of overall federation purpose, expected sequences of interactions, which federates that are producers and consumers of certain data, networking, etc.

2.1 Content of a FOM

The FOM can contain data in a number of different tables. A typical starter FOM usually includes the following tables.

Identification table, describing things like the purpose, author and version of the FOM.

Object Classes and Attributes tables, describing the persistent entities (like aircrafts) that are shared between the federates

Interaction Classes and Parameters tables, describing short-lived data like commands or radio traffic (i.e. events) that needs to be exchanged.

Data Types table describing the technical format and interpretation of the attribute and parameter data.

More advanced FOMs may also include the following:

Additional Federation Execution Data, like the **Dimensions table** (for DDM data routing), **Synchronization Points table** (for synchronizing the federation) and **User Supplied Tags table** (specifying the data format of extra parameters in certain HLA Services).

Additional Infrastructure Settings like the **Time Representation table**, the **Update Rate table**, the **Transportation Types** table and the **Switches table**.

For additional documentation across the tables it is also possible to attach a number of notes using the

Format	Technical format	Specified using	New features
HLA 1.3	BNF (”LISP” style)	BNF + textual specificatition	-
1516-2000	XML	DTD + textual specification	Routing Spaces table replaced with Dimensions. New table: Data types.
1516-2010	XML	Three XML Schemas + textual specification	Modular and extendable format. New tables: Update Rates, Services Utilization. Extended tables: Transportation Types, Identification

Figure 1: The evolution of the HLA OMT format

Notes table. There is also a **Services Utilization Table**.

2.2 Evolution of the FOM format

The FOM follows the OMT format of HLA. This format has evolved over time as shown in Figure 1. Two major driving factors can be noticed. First of all, new tables have been added or existing tables have been extended to meet new technical requirements. Secondly, the technical format has gradually been adapted to follow the most recent XML format descriptions.

2.3. Maintaining a FOM for multiple HLA standards

It is possible to convert FOMs using the older format to a newer format with automated tools. In some cases design decisions needs to be made, like mapping Routing Spaces to Dimensions or splitting a FOM into modules. This can technically be solved with a wizard, allowing for user input during the conversion process.

When maintaining a FOM for multiple standards it may be more convenient to keep the original FOM in a newer format since this is in many respects a superset of the older format. Data can then be converted back and new types of information can simply be excluded.

3. FOM Modules

The suggestion to make the FOM modular was the last comment in the last SISO comment round for the new HLA 1516-2010 standard. Still this idea had been around and discussed since the very first OMT specification in the 90’s.

The Modular FOM approach is very simple. Each FOM module describes a certain aspect of the information exchange. A FOM module can contain whatever FOM data that is required for its purpose, for example just a few object classes with attributes and corresponding data types. Several FOM modules can be combined into the final FOM to meet the requirement of a federation. You may for example combine a module describing vehicles with other modules describing radio communication, federation management and data logger control. The FOM data from these modules are then merged producing the union of the modules.

FOM modules make the development and reuse of FOM data more powerful. In many cases it makes FOM development easier since different development groups or parts of the development cycle can focus on particular areas of the FOM development.

For an extensive introduction to FOM Modules the paper “Getting Started with FOM Modules” [11] is strongly recommended.

3.1 Use cases for FOM modules

As described in [11], FOM modules can be used for several purposes:

You can have **different working groups** develop different parts of a FOM in a more convenient way. You may for example have a radio specialist group develop the “Radio FOM module” while the aircraft specialists develop the “Aircraft FOM module”.

You can put **extensions to a reference FOM** in a separate FOM module. This will prevent you from getting modified, “non-standard” versions of the reference FOMs. More importantly, when you build new federations using federates that use extended reference FOMs, it is easy to inspect what extensions that have been made and possibly to merge them.

You can achieve **extended reuse** of some aspects of a FOM. If you want to promote a standardized way to start and stop all of your federations, irrespective of domain, you may put these interactions in a separate FOM module.

You may also **add more FOM modules to an already executing federation**, thus extending the scope of the FOM during runtime.

Restaurant FOM	Sample FOM that is included in the HLA standard, available in HLA 1.3. 1516-2000 and 1516-2010 format. This FOM has been modularized as a sample in the product described below. It will be made publicly available on the SISO HLA Evolved PSG reflector in Sep 2010
RPR FOM	FOM mainly targeted at real-time platform simulations, standardized by SISO, matches the DIS data model. It is a good candidate for splitting into FOM modules. Some early work on modularization has been done as part of the BOM [12] standard, as described in another paper [13]
LINK-16 BOM	The SISO LINK-16 BOM is essentially a FOM module for LINK-16 [14] data exchange.
NASA	NASA has performed some early FOM module prototyping for the Constellation program as described in another paper [15]
P2SN FOM	Persistent Partner Simulation Network, previously Partnership for Peace Simulation Network. Originally in HLA 1516-2000 format, now in HLA 1516-2010 format. Uses the RPR FOM as one module and has both more general and more specialized FOM modules as described in another paper [16]
NETN FOM	NATO Education and Training Network FOM is expected to supersede the P2SN FOM above and will thus be modular.

Figure 2: Some efforts related to FOM modules

3.2 Some early modular FOM efforts

The HLA Evolved standard has recently been completed but there are already some early use cases for the modular FOM and some related efforts. Some examples are shown in Figure 2.

4 Processes for FOM module development

From the point of view of a particular federation the FEDEP/DSEEP [17] process describes at what stage the FOM needs to be developed. In this paper we look at the overarching perspective of reusing FOM modules across projects. This is further divided into how reusable FOM modules are produced and how they are later reused in other projects.

4.1 Developing standardized and reusable FOM modules top-down

Assume that a certain community wants to increase the potential for reuse of interoperable simulators. One of the most important efforts towards that goal is then to agree on a common, standardized FOM module. Since requirements will vary from federation to federation a good starting point is to include the most common shared objects and interactions in their domain. More specialized aspects can be covered in project-specific FOM modules. As many simulators in the community are adapted to the common FOM module they will have a basic level of interoperability.

Such FOM modules can be developed on several levels as shown in Figure 3.

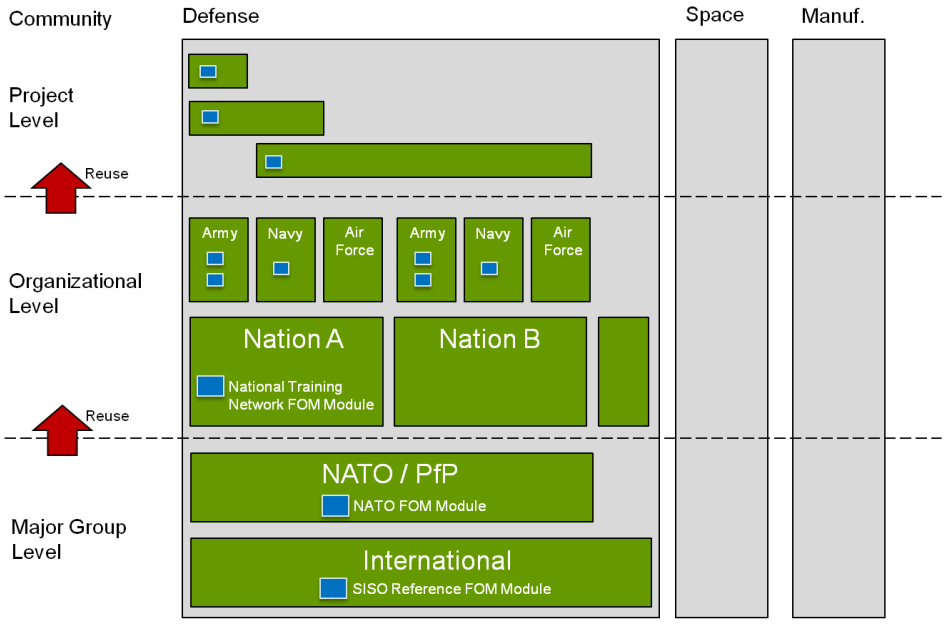


Figure 3: Developing reusable FOMs top-down

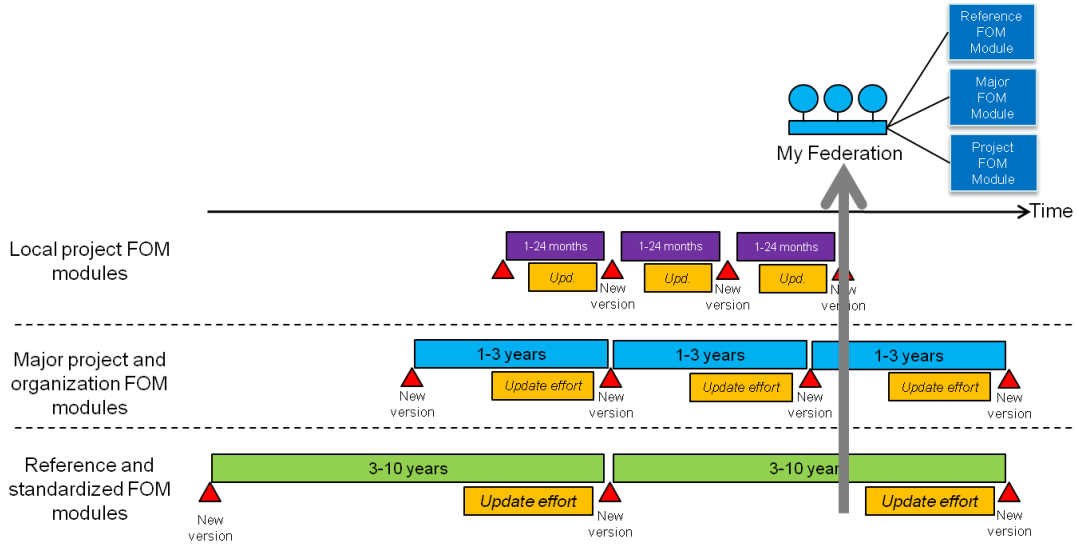


Figure 4: Reusing FOM modules with different lice cycles

On the overall community or domain level (defense, space, energy, manufacturing, etc) FOM modules can be developed in open international standardization forums, like SISO. The RPR FOM is an example of a FOM in the earlier HLA formats that is developed this way. Major groups with common needs, for example NATO, may also choose to develop common modules.

On more specific levels reusable FOMs can be developed within nations, within defense components or within companies.

Some important success factors when developing FOM modules are:

A clearly defined goal and a common requirements picture for the FOM module to be developed. Otherwise a consensus on a solution may never be reached.

A well-defined development process where new versions are developed, documented and released in a clear process that is transparent to the participants. Otherwise numerous of intermediate and ill-understood versions of the FOM will be used, actively preventing rather than enabling interoperability.

Management support for the development activities, for participants and for the resulting FOM module. If this cannot be achieved the development resources may be under-critical and the FOM will never be finalized or put into production.

Use-case and test-driven development. Practical experiences show that a suggested FOM solution that looks good on paper may contain minor glitches that makes it hard or impossible to use. Practical tests are the only solution to this. This process is unfortunately seldom documented in papers. An example from the DIS world of how a data exchange model has been adjusted after testing is the Directed Energy PDUs [18].

4.2 Developing reusable FOM modules bottom-up

It is also possible to develop reusable FOM modules from a more technical "bottom-up" perspective. Useful components like FOM elements are sometimes reused within and across federations, often even beyond the planned life span. The main reusability criteria here is usually "the survival of the fittest".

4.3 Reusing FOM modules

When reusing FOM modules in a federation a developer will typically start with some reference FOM modules. These can then be extended with locally produced FOM modules and extended with some project-specific FOM modules. In this case dependencies will typically be introduced. It is now important to look at the life-cycle of different modules. As shown in Figure 4, reference FOM modules will probably have a version life-cycle in the range of years whereas the project may update a FOM module every month. Proper configuration management of FOM modules is strongly recommended here.

4.4 Resulting requirements for a FOM module tool

Some important requirements for a FOM module editing tool that can be derived from the above are:

1. A tool should make it easy to **combine** different FOM modules that cover different information exchange requirements of the federation. It should be easy to get an **overview** of the combined modules and to **discover, analyze and fix any mismatches.**

2. A FOM module tool should enable a developer to **build upon standardized FOM modules**,

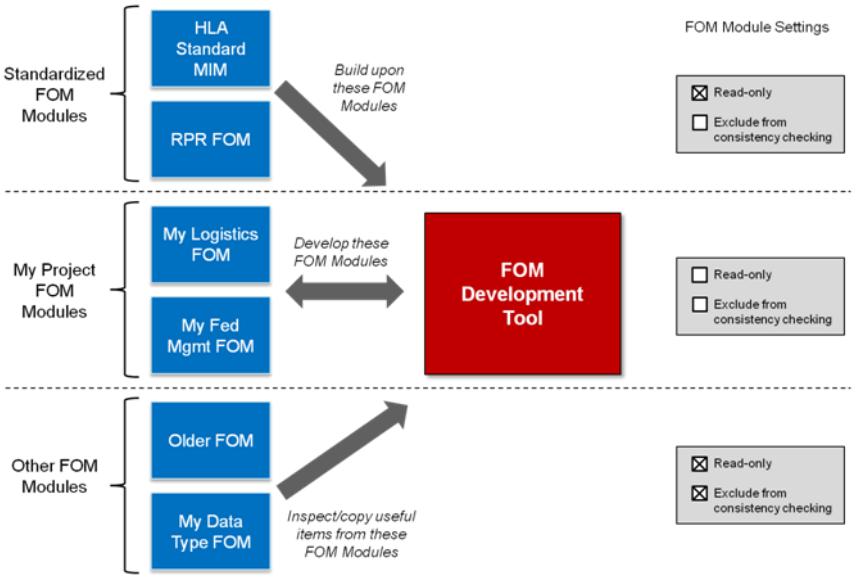


Figure 5: Composing a FOM from existing modules

add his own local extensions and also to reuse components from other, less related projects, as shown in Figure 5.

3. Each FOM module designer need to clearly understand which of the FOM modules in a project that he is responsible for updating and which that are maintained by other designers. A tool should support the developer in **updating only his own modules** and keep other read-only.

4. It is important to understand which FOM modules that are allowed to depend on others modules and which these modules are. A FOM module tool should assist in **maintaining proper dependencies**.

5. In many cases a user wants to work with existing modules just to copy data types or object classes that are generally useful. This module may never be intended to be used in the final FOM. A FOM module tool should make it easy to **reuse smaller components of older FOMs** that in their entirety may be less reusable.

5. A Next Generation FOM editing tool

A commercial tool for developing FOM modules, Pitch Visual OMT version 2.0, has been developed based upon the above analysis. It builds upon older versions that supported HLA 1.3 and HLA 1516-2000 FOMs. This versions is a complete reimplementation with HLA 1516-2010 FOM modules as the native file format. Still it supports many file formats of older HLA versions. Pitch Visual OMT 2.0 is planned to ship September 2010.

The overall design resembles a traditional graphical programming environment as can be seen in Figure 6. A project is composed from several FOM modules, including a standard or custom HLAstandardMIM. These modules are listed to the left and en-

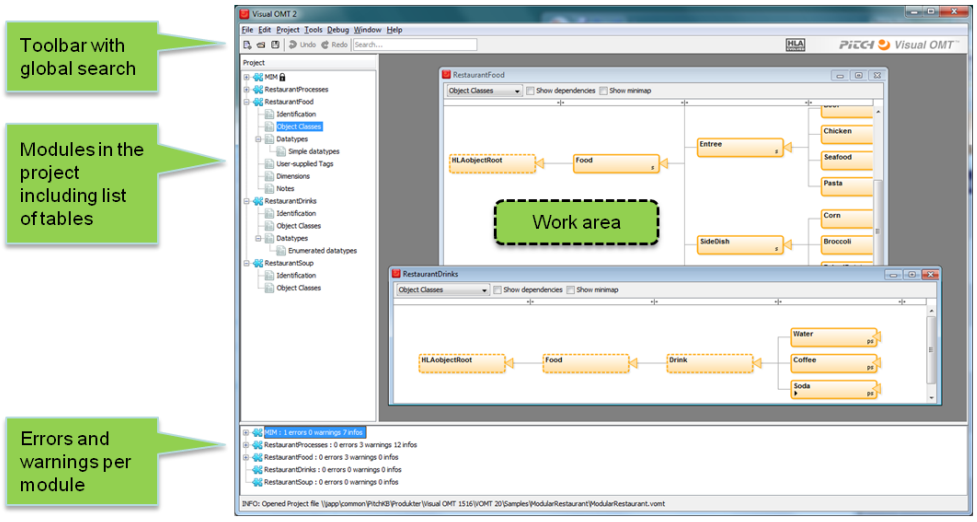


Figure 6: Overview of the Visual OMT 2.0 user interface

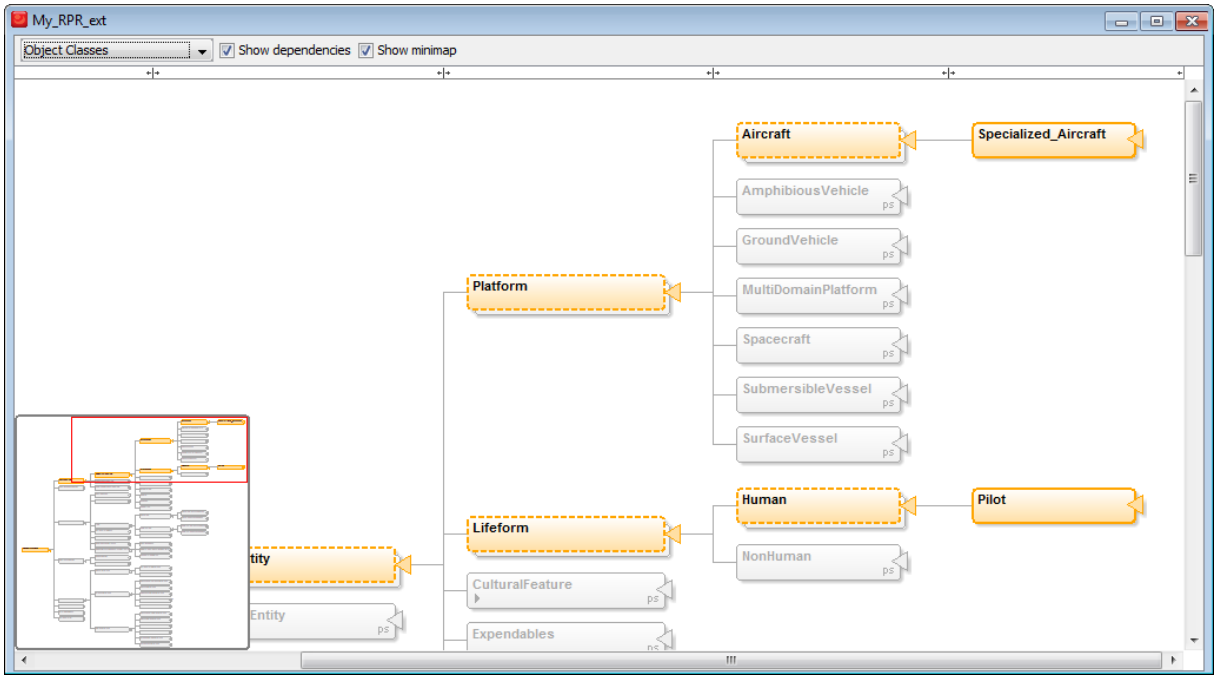


Figure 7: Object class view in Visual OMT 2.0

ables the user to inspect which tables that are included in each module. A padlock icon indicates that a module is read-only.

FOM modules are opened in the central work area. By double-clicking on a particular table in a module a developer can jump directly to that table, inspect FOM data, edit and use drag-and-drop.

The entire project is continuously analyzed in the background. Issues (errors, warnings and tips) are presented in the bottom area as a "To Do" list. There is a traditional toolbar at the top. One of the most interesting features here is the Global Search, as described later in this paper.

5.1 Visualizing combined FOM modules

Most FOM modules contain classes and subclasses for Object Classes and Interactions. The combined modules thus may result in a very large class hierarchy. The classes of each FOM or a set of dependent FOMs can be visualized as a tree graph in Visual OMT.

Modularized FOMs build upon each other and often extends classes in reference FOMs. The newly created subclass in such an extending FOM will have all its super classes defined as scaffolding classes. A scaffolding class is a way for a FOM designer to place a new class inside an existing class hierarchy without the need to duplicate all the super classes inside his FOM module. This allows the designer to see the whole chain of objects from HLAobjectRoot to the new class.

However, when looking at only one module it is not possible to know if a class in the current module is also defined in another module, or if the class inherits attributes from a super class in another module. Classes exist in a context dependent on which module it is supposed to merge with. This concept is hard to grasp while only looking at one module at a time. By combining several class trees and displaying them together a more comprehensive view of the context of the classes is created. In a combined view it is easier to see where to place new classes and to see how it fits with already existing classes.

Figure 7 depicts such an combined class tree structure. The colored classes belong to the module currently being edited and the grey classes provide context information. When there are multiple definitions of a class in several modules it is shown as a stack. The dashed lined denotes a scaffolding definition. Here it is easy to see if a scaffolding class has a proper full definition in another module and if any of the definitions clash with other modules. Editing is simplified by allowing drag and drop to rearrange classes. Making it easy to maintain consistency with other modules is also enabled by supplying functionality to copy a full definition to a module and create new classes that automatically get the appropriate scaffolding classes to connect it with the combined class hierarchy.

5.2 Locating information

Working in large FOMs, or rather with a collection of FOM modules where the resulting FOM is large, there are several features that can simplify navigation. In Visual OMT 2.0 there is a global search function which make it possible to search the project in order to find a certain type or concept. An

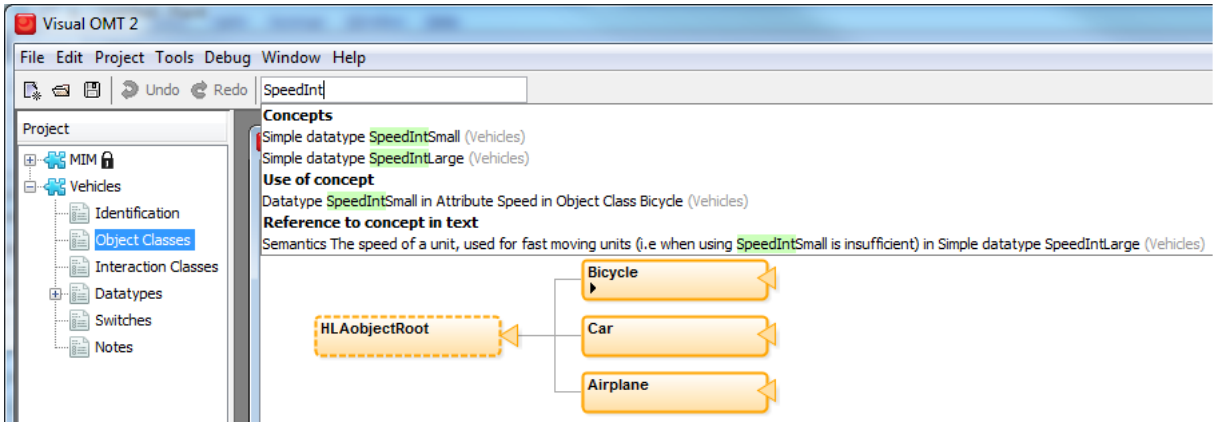


Figure 8: Example of a global search in Visual OMT 2.0

example of a global search can be seen in Figure 8. It is possible to double click on an item in the list to go to its corresponding table. There is also a specialized datatype search used to find the datatype appropriate for attributes and other concepts that uses datatypes. In this search it is possible to sort the resulting datatypes on name, type and other characteristics

To ease navigation in large class trees there is a mini-map showing which part of the graph is currently displayed, visible in the lower left corner of Figure 7. The mini-map also allows for quick navigation. In the main view there is also the possibility to expand and collapse sub-trees to look at only parts of the tree.

5.3 FOM dependencies

When working with FOM modules it is often a good idea to gather information that many FOM modules need in a single FOM module. This single FOM module would contain datatypes, dimensions and other objects shared among the FOM modules. This introduces a dependency between the different FOM modules in which one FOM module is dependent on another and requires that it also is used in a federation.

Another situation which would create a dependency is when a FOM designer uses a reference FOM, such as the RPR FOM, to create a FOM module that extends the reference FOM.

Visual OMT 2.0 helps a FOM designer by allowing him to explicitly define allowed dependencies and then enforces them. Any usage of objects, such as a data type, that's not defined in the module itself, or its dependencies, will generate a warning prompting the user to either define the dependency explicitly or to use another data type.

The concept of dependencies also reduces the amount of data to process when editing large FOMs, since data from modules that are not specified in dependencies will not show up as alternatives when choosing data to use in the new module.

5.4 Read only FOM modules

In larger projects the responsibility for creating the FOM is often split up amongst a group of people so that each person creates a FOM module that together forms the final FOM. This means that from the perspective of one person the FOM modules he is not responsible for should be considered read only, as he lacks the authority to change them. Another case in which a FOM module should be considered read only is when developing an add-on module to a reference FOM such as the RPR-FOM, in which case the RPR-FOM should be considered read only. In cases such as these Visual OMT 2.0 allows the FOM designer to mark a module as read only. A read only module may not be modified but FOM modules that depend on it are allowed to make use of the data in it as described earlier.

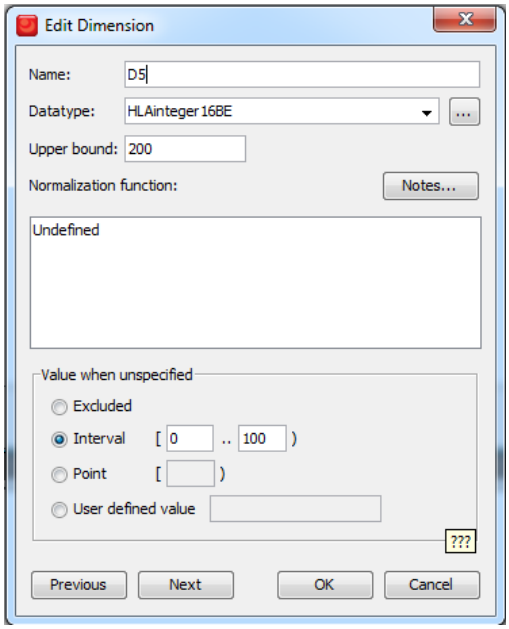
5.5 Sample FOM Module projects

Sample projects makes it easy to get started and demonstrate how different parts of a FOM are defined and how they work together. It also gives examples of how functionality is distributed among the modules. Some examples included in Visual OMT 2.0 are:

Hello Modular World: a very basic sample

Modular Restaurant: based on the sample Restaurant FOM in the HLA Standard. This example illustrates both how one module (Soup) can build upon another module (RestaurantFood) as well as how two independent modules, in this case RestaurantFood and RestaurantProcesses can stand side by side.

Sample RPR FOM Extension. This example may be more difficult to grasp for someone who is unfamiliar with the RPR FOM. Still it represents an example of how many real-world developers would use FOM modules.




```
<dimension>
    <name>D5</name>
    <dataType>HLAinteger16BE</dataType>
    <upperBound>200</upperBound>
    <normalization>Undefined</normalization>
    <value>[0..100)</value>
</dimension>
```

Figure 9: Editing with a dedicated tool versus editing text format

5.6 Help and explanations

A great advantage of a specialized editing tool as opposed to general XML editors is that more support can be added to help the user enter correct input. Even though XML editors can contain some support, there are several things they cannot check and they can never give more help than there is in the XML schema. A specialized tool like Visual OMT 2.0 has "tooltips", help and explanations as well as GUI components that simplifies editing. Instead of having to know by heart which data types there are to choose from, a drop down box show the alternatives. The GUI also help with advanced formatting like Dimension default values, see Figure 9.

5.7 Locating errors in a FOM module

An easy mistake to do when creating a FOM module is to create references to non existing objects. For example, by misspelling a data types name or defining that an attribute should use a data type that has not yet been defined and later on forgetting to do so. In Visual OMT 2.0 the FOM designer is presented with a list of already defined data types when selecting a data type for an attribute or selecting the dimension for an interaction class. This also incorporates the dependency system described in an earlier section. When selecting a data type for an array, not only data types in the current module will be available, but all data types in the current module and its dependencies

As mentioned earlier an easy mistake to do when creating FOM modules is to create a reference to a non existing object. This is of course not the only kind of error that can occur in a FOM module. Visual OMT 2.0 can identify almost 200 different types of issues. These have been divided into three different groups, errors, warnings and best practice. An error is classified as something that would make the RTI unable to successfully load the FOM module such as two object classes with the same fully qualified name but with different definitions. A warning is something that is not correct but the RTI will still be able to load the FOM module, for example a reference to a non existing data type in an attribute. Finally best practice is tips for creating a good FOM, this includes filling out the Identification table and defining semantics for all your objects.

Visual OMT 2.0 continuously analyzes the FOM modules in the project. When issues are found, they are presented in a to-do list at the bottom of the screen as shown in Figure 10. This gives the FOM designer an easy way to identify how many errors exist in the FOM modules as well as a way to quickly go to an error by double clicking on it in the to-do list.

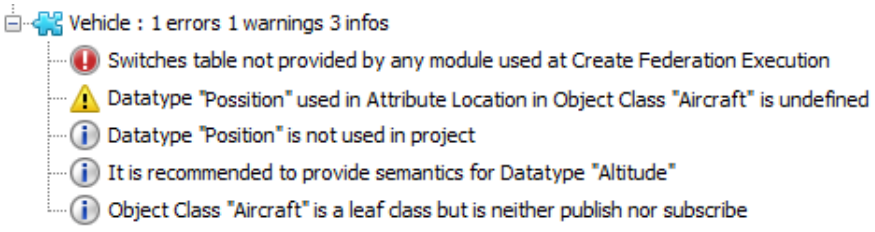


Figure 10: Example of the issues identified in a project in Visual OMT 2.0

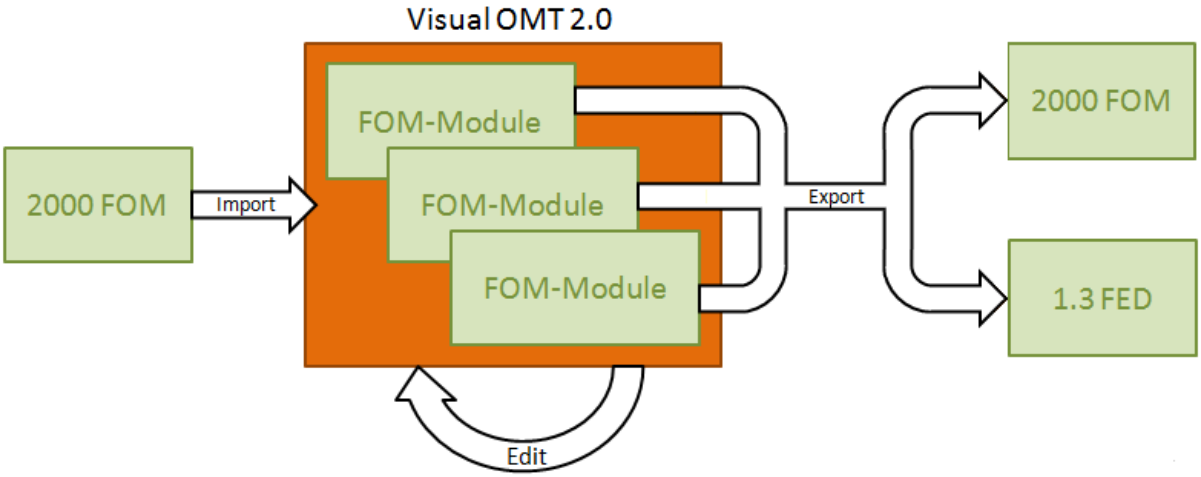


Figure 11: Handling different HLA formats

5.8 Help with module design

Another way in which a FOM editing tool can help a FOM designer is by suggesting good default values. Not only default values for specific things such as the order type of an attribute but design choices like which tables is normally best practice to define in a FOM module. Default values can be controlled per project so that a training oriented FOM uses best-effort transportation by default whereas an analysis FOM defaults to reliable transportation and time stamp order delivery.

Visual OMT 2.0 will also always create an Identification table in a FOM module, although it won't force the FOM designer to actually enter anything useful.

5.9 Reusing FOM data

Reuse of FOM data in a non specific FOM editing tool can be tricky, often involving copy of raw text where care has to be taken to not miss an XML tag and correctly paste the information in the new FOM.

Reusing elements from older FOMs is simple in Visual OMT 2.0. It is possible to open the module and drag and drop entities like object classes or data types to other modules.

If you want to have an older FOM in your project to consult as a reference you can choose to exclude it from the consistency checking in the project since such a FOM probably would introduce multiple errors if combined with the other FOMs.

5.10 Ensuring valid HLA 1516-2010 File format

A fundamental requirement on a FOM editing tool is that it should produce a valid FOM file. A FOM file has a valid OMT syntax if it validates against the DIF XML schema defined by the HLA standard.

In Visual OMT 2.0 this is ensured by using a software component for reading and writing FOM files which is based on this Schema. This also means that Visual OMT 2.0 easily can handle any changes to the FOM format by updating the software component.

5.11 Handling older HLA formats

With HLA Evolved and two older HLA versions now available it is often the case that a FOM needs to be maintained for use in federations using different HLA versions. In Visual OMT 2.0 this can be solved by maintaining one set of FOM modules in the latest HLA Evolved format and then exporting them to older formats. Export to both HLA 1516-2000 and to HLA1.3 FED allows you to run any version of the RTI. An overview of this approach can be seen in Figure 11.

Visual OMT 2.0 also has the ability to import old FOMs from HLA 1516-2000 meaning you can rearrange old FOMs into modules making them easier to understand and maintain.

5.13 Future development

One feature we are looking at adding to Visual OMT 2.0 in the future is refactoring. This would, for example, allow a FOM designer to change the name of a data type and the new name would propagate to all instances where the old name was used.

We are also planning to add more analysis for detecting issues within a project. This could also lead to Visual OMT 2.0 being able to supply the user with a suggested method of resolving an issue and then carry it out if the user accepts it.

More convenient methods for importing and exporting data will also be added. One example is the ability to paste a column of data from Excel into Visual OMT 2.0. This is very useful when defining enumerator values for a new enumerated data type.

Feature	Text Editor	XML Editor	Visual OMT 2.0
Ensures correct syntax	No	Yes	Yes
Help with semantics	No	No	Yes
Ensures correct references	No	No	Within and between modules
Checks best practice	No	No	Yes
Graphical visualization	No	General XML format graph	Object and Interaction class trees
Tabular visualization	No	No	Similar to the HLA standard
Dependency analysis	No	No	Yes, searchable
Search	In one module	In one module	Across several modules
Import/export between formats	No	No	Yes

Figure 12: Comparison between FOM editing options

5.12 A comparison with other editors

Figure 12 summarizes some functional differences between editing options. In addition to this cost may also be considered. A basic text editor like Notepad will of course be a cheap alternative for small projects. Larger projects will want to opt for a dedicated HLA OMT editing tool.

6. NATO Snow Leopard example

This section describes a project where FOM modules are used. It gives a project overview and then shows an example of how a real FOM module looks in Visual OMT 2.0.

6.1 Project overview

The NATO Snow Leopard a.k.a. NATO Education and Training Network (NETN) is based on recommendations and federation agreements developed by MSG-068 (NATO RTO Task Group). These agreements include a set of FOM modules that use and extend existing standard object models, e.g. RPR-FOM and Link 16 BOM.

The basis for the development of the NETN FOM based on this set of modules are national and NATO federations and simulation systems including NATO Live-Virtual and Constructive Federation (NLVC), NATO Training Federation (JTF), Joint Multi Resolution Model (JMRM), German KORA-SIRA (KOSI) federation, Persistant Partner Simulation Network (P2SN) federation, French ALLIANCE Federation, and other simulation systems from Spain, UK, The Netherlands, Australia, Bulgaria, Romania, Turkey, USA and Sweden.

Early in the development of the NETN Reference Federation Agreements it was decided to base the FOM on standards, best-practices and practical experience from the participating nations and organizations. A driving factor in the design of the FOM was to enable a higher level of interoperability between the systems and to allow the use of national simualtion systems in NATO Computer Aided Exercises (CAX).

6.2 Use of FOM modules

The PRP-FOM v2.0 D17 was selected to form the basis for representing Ground Truth for both Platforms and Aggregate Entities. To allow additional information to be exchanged between systems the Platform and Aggregate Object Classes of the RPR-FOM were extended by adding a NETN-Aggregate FOM Module. This module subclasses all the platform object classes and the RPR-FOM AggregateEntity object class.

Another FOM Module called NETN_Logistics was introduced to allow more advanced cross-fedederate logistics operations. It also includes NETN_Facility as a new subclass of BaseEntity. Although RPR-FOM include some basic support for logistics MSG-068 recommendation is to completely replace this with a new way of requesting and providing logistics services. The more general "Service Consumer-Provider" pattern was captured in a separate FOM module and extentions for logistics services was put in another Module. The will allow future modules to utilize the same basic pattern for services whithout including the Logistics module.

The NETN Reference FOM includes the following modules:

- RPR-FOM v2.0 D17
- Link 16 BOM
- NETN Service Consumer-Provider

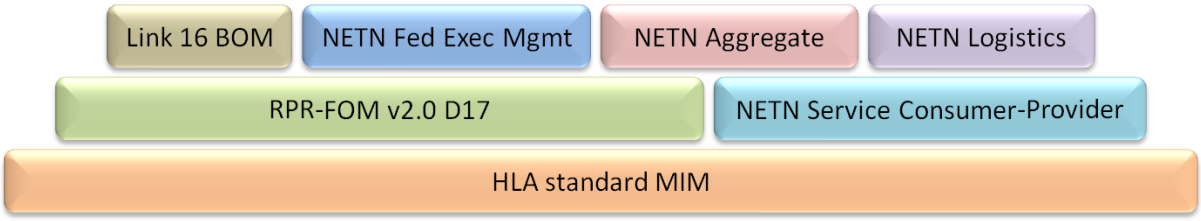


Figure 13 Visualization of dependencies between FOM Modules in the NETN Reference FOM

- NETN Logistics
- NETN Aggregate
- NETN Federation Execution Management

The relationship between the different modules can be seen in Figure 13, a module is dependent on the modules below it in the diagram.

6.3 Logistics FOM module interactions

Figure 14 shows the Interaction class tree for the NETN Logistics FOM module as it is displayed in Visual OMT 2.0. It is easy to see that for example the NETN_SupplyStarted interaction extends an interaction class that is defined in another module, in this case NETN Service Consumer-Provider.

7. Discussion

Some thoughts and challenges that came up during the development of the tools are summarized here.

7.1 Collaboration and the FOM development process

Early in the tool design and experimentation phase we realized that most FOM module editing needs to be focused around a project consisting of several modules. The project typically contains some modules that need to be read-only, from the perspective of a particular federation. An example of this is when several developers share FOM modules under development with each other. When a conflict occurs it is necessary to adjust at least one of the conflicting modules. In some cases the issue to be resolved may be in a read-only FOM, often perceived as "somebody else's" module. In this case it would be a good idea to generate a "change request" from within the tool, instead of updating a read-only module. This is considered for inclusion in later versions of the tool.

On a higher level the entire collaborative process of FOM development may be supported by web based tools. Typical tasks in such an environment would be to handle suggestions and change requests as described in the previous section. This may be a future development.

7.2 Ease of use versus advanced features

It is a challenge to design a tool that both correctly covers the entire standard and, at the same time, is easy to use for beginners. One example of this is how the MIM is handled in Visual OMT 2.0. For the beginner the MIM in itself is not interesting, he probably just want access to the predefined data (such as datatypes). To make it easy for a beginner, the standard MIM is automatically added to all projects and all modules are by default defined as dependent upon it. If ease of use was the only design goal this functionality would be enough, but an advanced user might want to use a non standard MIM, which is also supported by the tool, including the related error-checking.

7.3 Using the OMT glyph as an icon

Visual OMT 2.0 displays the Glyph of the Identification table as an icon in the FOM module overview screen. One issue here is that the Glyph can be any size and any image format. It would be desirable to further standardize the Glyph concept in the Identification table. This would allow programs

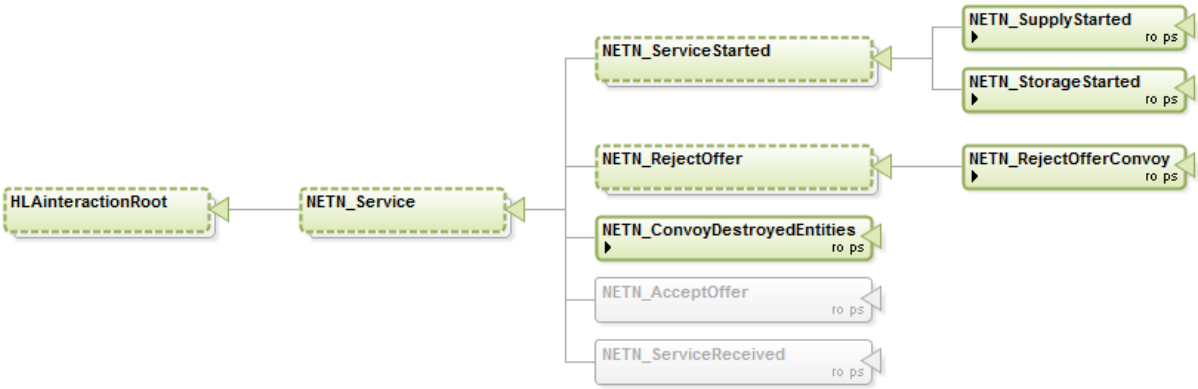


Figure 14: Part of the Interaction class tree for the NETN Logistics FOM

that handle FOMs to use the glyph as an icon for a FOM in their GUI. We recommend standardizing on the GIF, JPG and PNG formats and the size 32 by 32 pixels.

7.4 Attaching Notes to any item

The Notes concept in OMT is difficult to implement in a straight-forward and user friendly way. The OMT format is specified using tables, making it natural to define that it should be possible to place a note on a column. XML is the format used to actually save a FOM file. It lacks the notion of columns. Overall the specified ability to place a note on anything, including placing a note on a note, may result in a complex GUI.

8. Conclusion

The FOM, i.e. the information exchange data model, which is used in a federation is extremely important from both an interoperability and reuse perspective. It is considerably easier to make simulations originally based on similar FOMs to interoperate. It is also more likely that federates can be reused in a federation with a FOM that is similar to their original information exchange data model. HLA Evolved makes it considerably easier to develop and reuse different aspects of a FOM by providing the FOM as composable modules.

Reusable FOM modules ideally should be developed and maintained using a clear and well-defined process. This can be in a top-down process by organizations sharing a common need to support a certain simulation domain. In some practical cases a really useful block of FOM data may be produced in a project and then reused on an ad-hoc basis.

In order to be able to develop FOM modules and to compose entire FOMs from modules, proper tools are needed. They should enable the user to get a clear overview of several combined FOMs, to find errors and to analyze and resolve conflicts between modules. Based on the role of each FOM module in a project the tool should assist a user in keeping standardized modules untouched while developing extensions. A tool should also assist a user in maintaining proper dependencies so that a stand-alone module remains stand-alone and a dependent module only depends on the intended modules. This paper describes how a COTS tool that supports this has been developed.

An example of more general and more specific FOM modules that have been developed in the NATO Snow Leopard federation are also provided.

To summarize, this paper shows how the FOM module concept can take interoperability and reuse to new levels, with the ease-of-use and convenience of a graphical FOM editing tool.

References

[1] IEEE: "IEEE 1278, Distributed Interactive Simulation (DIS)", www.ieee.org,

[2] Vinton Cerf: “Specification of Internet Transmission Control Program”, RFC 675, IETF, www.ietf.org, December 1974

[3] J Postel, J. Reynolds: “File Transfer Protocol (FTP)”, RFC 765, IETF, www.ietf.org, October 1985

[4] R. Fielding et al: “Hypertext Transfer Protocol - HTTP/1.1”, RFC 2616, IETF, www.ietf.org, June 1999

[5] Jonathan B. Postel: “Simple Mail Transfer Protocol”, RFC 821, IETF, www.ietf.org, August 1982

[6] IEEE: "IEEE 1516-2010, High Level Architecture (HLA)", www.ieee.org, To be published.

[7] “TENA - The Test and Training Enabling Architecture, Architecture Reference Document”, https://www.tena-sda.org/public_docmanager/userdocuments/TENA%20ARCHITECTURE%20REFERENCE/TENA%20Architecture%20Reference%20Document%202002.pdf

[8] Thomas Erl: “Service-Oriented Architecture, Concepts, Technology and Design”, Prentice-Hall, July 2005, ISBN 0-13-185858-0

[9] Object Management Group: “Data Distribution Service for Real-time Systems, v1.2”, www.omg.org, January 2007.

[10] MIL-STD-6016B, Tactical Digital Information Link (TADIL) J Message Standard (DRAFT) 15 March 2002

[11] Möller, B and Löfstrand, B, “Getting started with FOM Modules”, Proceedings of 2009 Fall Simulation Interoperability Workshop, 09F-SIW-082, Simulation Interoperability Standards Organization, September 2009.

[12] SISO, “BOM Template Specification”, SISO-STD-003-2006, SISO, www.sisostds.org, 31 March 2006.

[13] Tram Chase, Paul Gustavson, Lawrence M. Root: “From FOMs to BOMs and Back Again”, Proceedings of 2006 Spring Simulation Interoperability Workshop, 06S-SIW-115, Simulation Interoperability Standards Organization, April 2006

[14] SISO, "Standard for: Link16 Simulations", SISO-STD-002-2006, SISO, www.sisostds.org, May 2006

[15] David Hasan: "Using HLA-Evolved Modular Object Models for NASA Constellation Simulation-to-Simulation Interface Specifications", Proceedings of 2010 Spring Simulation Interoperability Workshop, 10S-SIW-012, Simulation Interoperability Standards Organization, April 2010.

[16] Björn Löfstrand, Rachid Khayari, Konradin Keller, Klaus Greiwe, Peter Meyer zu Drewer, Torbjörn Hultén, Andy Bowers, Jean-Pierre Faye. "Logistics FOM Module in Snow Leopard: Recommendations by MSG-068 NATO Education and Training Network Task Group", Proceedings of 2009 Fall Simulation Interoperability Workshop, 09F-SIW-076, Simulation Interoperability Standards Organization, September 2009.

[17] IEEE: "IEEE 1516.3-2003 IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP)", www.ieee.org

[18] Joe Sorroche, Riley Rainey: "Directed Energy Modeling and Simulation Experiment Results", Proceedings of 2007 Spring Simulation Interoperability Workshop, 07S-SIW-042, Simulation Interoperability Standards Organization, April 2007.

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. He is currently serving as the vice chairman of the SISO HLA Evolved Product Development Group.

FREDRIK ANTELIUS is a Lead Developer at Pitch and is a major contributor to several commercial HLA products. He holds an MSc in computer science and technology from Linköping University, Sweden.

MARTIN JOHANSSON is Systems Developer at Pitch Technologies and is a major contributor to several commercial HLA products such as Pitch Developer Studio and Pitch Visual OMT 2.0. He studied computer science and technology at Linköping University, Sweden.

BJÖRN LÖFSTRAND is the 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 federation architecture and design patterns for HLA based distributed simulation. He leads the MSG-068 FOM and Federation Design (FAFD) technical subgroup.

ÅSA WIHLBORG is Systems Developer at Pitch Technologies and a major contributor to commercial HLA products such as Pitch Visual OMT 2.0. She studied computer science and technology at Linköping University, Sweden.