Making Your BOMs and FOM Modules Play Together

Björn Möller
Paul Gustavson
Bob Lutz
Björn Löfstrand

bjorn.moller@pitch.se
pgustavson@simventions.com
robert.lutz@jhuapl.edu
bjorn.lofstrand@pitch.se

Keywords:
Interoperability, Modeling, BOM, FOM Modules, FEDEP, HLA, RTI

ABSTRACT: *Proper modeling is key in order to achieve effective interoperability between simulation systems. Base Object Models (BOMs) is a SISO standard that allows simulation developers to create object models that form a base for interoperability, even though the participating systems to be used have not yet been selected. The BOM concept is based on the assumption that piece-parts of models, simulations, and federations can be extracted and reused as modeling building-blocks or components. Special attention is paid to sequences of events that take place between simulation elements.*

An important part of the High Level Architecture standard is the Federation Object Model (FOM) that describes the data to be exchanged at runtime. The upcoming version of HLA, named "HLA Evolved", allows FOMs to be divided into smaller, reusable components called FOM modules.

BOMs have unique capabilities in the earlier phases of FEDEP since they enable reuse across federations and have little dependency on the exact systems that are used in any particular federation. FOM Modules on the other hand have unique capabilities during the later phases since they can provide plug-and-play reuse. The greatest benefit is achieved if they are used together. BOMs and Modular FOMs also share a number of description formats that enables a smooth transition from BOMs to FOM Modules.

This paper describes in detail how BOMs and Modular FOMs can be used together for optimal reuse and interoperability from the early modeling stages to the final integration and execution phases.

1. Introduction

Simulation can be used for many purposes. It enables us to study and understand theories, concepts and processes. It allows us to make predictions about the future. It enables us to study the performance of organizations or equipment that may not yet exist in reality. It allows us to train and be part of virtual scenarios that would otherwise be too expensive or dangerous.

1.1 Simulations are based on models

No matter for what purpose simulation is used it is always based on models of an actual or anticipated reality. Creating such models is a non-trivial task. While reality, down to the level of atoms and molecules, is infinitely complex, models must be based on simplifications of the world. Anything else would be too time-consuming to develop, too expensive and too complex to execute. Still these simplifications must result in a model that is relevant for the purpose of the simulation.

The effort that it takes to develop a good and relevant model should not be underestimated. It usually requires the participation of people with highly specialized knowledge as well as motivation. If we want to simulate complex and real-world concepts we may need to get models from different domains and communities and make them interoperate. How different types of models can be captured, structured and reused is one of the most critical questions for the modeling and simulation community.

This paper is about the purpose, development, syntax and semantics of standards-based models targeted at interoperability and reuse.

1.2 Types of models over the life-cycle

Different types of models can be used over the life-cycle of one or more organizations modeling activity.

Early stages of modeling may require starting from an **ontology**[1] where the meaning of different concepts can be captured in detail. In many cases this step is skipped or a common understanding of the concepts is assumed.

The type of model that comes to most peoples minds is the **conceptual model** of a world. It would typically contain concepts, relationships and processes or state-changes over time. Good conceptual models have a big potential for reuse while a particular implementation may not always be reused.

One method towards developing a conceptual model may be to look at cases or scenarios. This is common practice for example in AI techniques such as knowledge acquisition. It is also a suggested technique in the FEDEP[2] standard.

If we have several types of models that are implemented in different computer applications we need to create an **information exchange data model** (IEDM). The information exchange data model may need to inherit some very specific properties from participating systems, such as data formats or shared algorithms, which makes the potential for reuse considerably smaller. The effort for reusing combinations of systems supporting the same information model is usually low, sometimes enabling what is popularly known as plug-and-play.

Another type of model is the model of a computer program. This is one of the fundamentals of the more successful parts of the Model-driven Architecture (MDA)[3]. In this case a Platform Specific Model (PSM), such as a C++ program, can be derived from a Platform Independent Model (PIM). In this case the structure and properties of the solution (executable) is modeled, not an actual or anticipated reality. Having mentioned that it is important to understand this difference; this type of model will not be further discussed in this paper.

1.3 Models and standards

If a model can be described in a standard format and using standardized building blocks the likelihood of reuse increases. If highly structured, machine-readable formats, like XML, are used even more opportunities open up for consistency checking, storage, manipulation, transformation and generation.

Recent standard development efforts within the SISO community have applied XML technology[4], whenever possible. In some cases, like HLA, earlier formats have been replaced by XML based formats. Tools are also available for automated conversion from earlier formats to XML based formats.

The Federation Development and Execution Process (FEDEP) is an IEEE Recommended Practice that provides a process framework for building a federation of interoperable systems. While currently focused on the High-Level Architecture (HLA)[5] the long-term goal is to support a wider range of distributed simulation interoperability standards.

Within SISO the Base-Object Model (BOM)[6] standard has also been developed. A BOM covers a wide range of the simulation development activities, from conceptual models to the building blocks for the information exchange model.

The High-Level Architecture (HLA) is more focused on the run-time aspects of a simulation. The main models of HLA is the Federation Object Model (FOM) and the Simulation Object Model (SOM). The FOM describes the data to be exchanged during the execution of a federation. Selected parts of the FOM are read by the RTI to initialize the federation execution. The SOM is a description of information that a simulation can exchange in a potential federation. The SOM is usually used for a preliminary evaluation whether a simulation is suitable for a particular purpose in a planned federation. The FOM has recently become more “module-based” as part of the HLA Evolved[7] effort.

FOMs and several parts of the BOMs have a very similar format. While the FOM is a pure information exchange data model the BOM covers several earlier steps in the process. The intent is not that FOMs and BOMs shall compete. They have different focus during the federation life-cycle as will be shown later in this paper.

BOMs are the tool of choice for the earlier development steps whereas FOM modules come in later. The common format enables a smooth transition from earlier, BOM-based, activities to later, FOM-based activities.

2. Description of HLA Object Models

The HLA FOM and SOM are based on the OMT format. A simple example of a shared “Vehicle” concept with an attribute called “Speed” is provided here.

```
<objectClass>
 <name>Vehicle</name>
 <sharing>PublishSubscribe</sharing>
 <semantics>Any vehicle</semantics>
  <attribute>
   <name>Speed</name>
   <dataType>SpeedInteger</dataType>
   <updateType>Periodic</updateType>
   <ownership>NoTransfer</ownership>
   <sharing>PublishSubscribe</sharing>
   <transportation>HLAbestEffort
        </transportation>
   <order>Receive</order>
   <semantics>Speed of the vehicle
        </semantics>
  </attribute>
</objectClass>
```

```
<simpleData>
 <name>SpeedInteger</name>
 <representation>HLAinteger32BE
       </representation>
 <units>mph</units>
 <resolution>1</resolution>
 <accuracy>5</accuracy>
 <semantics>Integer used for speed data
       </semantics>
</simpleData>
```

Figure 1: OMT Format Example

The exact format of a FOM is described in the OMT part of the HLA standard. The following is an informal overview of what a FOM contains:

- **Identification**. This is a general description of the FOM, domain, purpose, developer and so on.
- **Object classes** and **attributes**. Shared object classes are described, for example Vehicle. This is an object class hierarchy similar to what is used in object oriented programming. We can introduce subclasses, for example an Aircraft 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. The older HLA 1.3 standard provides predefined groupings called routing spaces whereas the HLA IEEE 1516 standard provides a more flexible set of dimensions that can be combined dynamically at runtime.
- **Data types** for attributes, parameters etc. HLA 1.3 provides a list of commonly used simple data types, like float, integer and string. The HLA 1516-2000 standard adds support for a full and unambiguous specification of data types down to the bit level, including complex data types, arrays with 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. It can be extended in case an RTI provides additional transportation types.
- **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 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.

There are a few additional things worth mentioning. There is an OMT table called **Lexicon** that describes the semantics of OMT data, for example attributes, parameters, etc. In the XML format as well as in many OMT editors they are integrated into the corresponding data objects instead of stored separately.

2.1 FOM Modules

In the most current HLA specification (IEEE 1516-2000, there is only one, monolithical FOM for each federation. When the federation execution is created, the RTI loads the FOM, enabling participating federates to publish and subscribe to FOM data. While the single FOM approach has been a useful means of establishing a valid data exchange "contract" among federates, it also restricts the speed, flexibility, and accuracy of the development process. This is the reason for the introduction of the FOM Module[8][9] concept as part of the HLA Evolved effort. With the FOM Module approach, the FOM can be broken down into several modules that may capture different parts of an object or interaction class hierarchy. This allows a subset of federates to specify what data they need to exchange upon joining, in addition to the initially loaded FOM. The main advantage to this approach is the increased flexibility during runtime and, not the least, during the development process. For instance, 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. With Modular FOMs, it is now possible to have long-running or persistent federations in virtual arenas where new capabilities and types of shared information can be added over time.

A simple example of this concept is shown in Figure 2. The blue HLAobjectRoot comes from a standard module for Management Object Model (MOM) information. The green entity classes (LifeForm, Human, Car) come from one FOM Module. The Italian concepts (Italian, Lamborhini, Ferrari), here shown in yellow, come from another FOM module. When the Italian federate joins, the FOM expands to include the information in these yellow boxes.

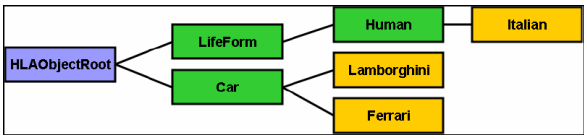

Figure 2 – FOM Module Example

3. Description of Base Object Models

The BOM is a recent Simulation Interoperability Standards Organization (SISO) Standard developed in the open community for the purpose of supporting composable and interoperable object modeling. It is defined as "a piece part of a conceptual model, simulation object model, or federation object model, which can be used as a building block in the development and/or extension of a simulation or federation."[6]

The idea behind BOMs actually can be traced back to the mid 90s when HLA was first being cultivated. It was then that this notion of a piece part concept was considered which could serve as building blocks in respect to the development process and the creation of interoperable object models[2].

A side-by side comparison of the BOM structure with the structure of a FOM is provided in Figure 3. The common areas between a BOM and FOM include the model identification component for describing the metadata of a model, and the ability to identify object classes, interaction classes and data types. A FOM also provides other components that the BOM need not represent. This includes HLA Dimensions, HLA Time, HLA Tags, HLA Synchronizations, HLA Transportations, HLA Switches. They are not part of the BOM because they were not seen as essential to document a BASE object model.

3.1 Conceptual Model Definition

The Conceptual Model aspect is one of the discriminators of the BOM; one of the things that sets itself apart. Prior to the BOM standard, the M&S community did not have a formal and easy way to describe and share conceptual model elements, and did not have an easy way to carry that conceptual model forward through the development process.

A BOM offers a formal way to capture and share the Conceptual Model, which is something not supported by a FOM. A Conceptual Model provides a description

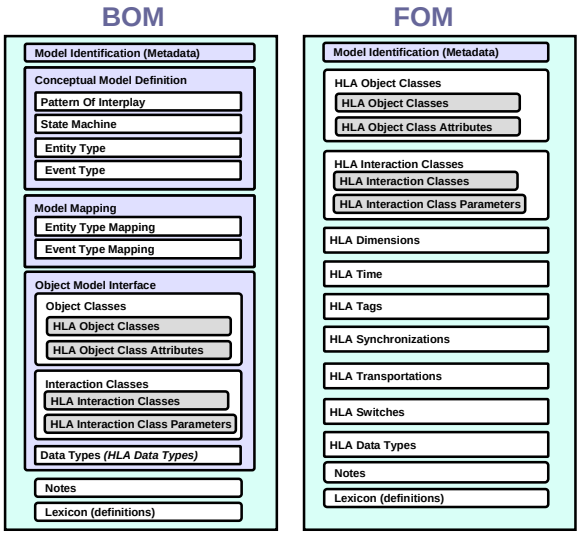

Figure 3 –
OMT Structure compared with the BOM Structure

of "what is to represented, the assumptions limiting those representations, and other capabilities needed to satisfy the user's requirements."[3] In regards to the conceptual model of a BOM, what can be reflected is the pattern of interplay, the states of an entity, the entity types and event types.

The idea of pattern discovery is very relevant to the conceptual analysis phase prescribed by the FEDEP. A Pattern is "an idea that has been useful in one practical context and will probably be useful in others." (Martin Fowler[10]). The Weapon's Effect pattern shown in Figure 4 is an example of something is done with some frequency in combat simulations; it is a pattern. This pattern approach is a differentiator from other object modeling frameworks. This aspect is important, because if intent can be understood as well as the anticipated behavior, then it is easier to know how to reuse something. The conceptual model forms the basis of defining reusable components.

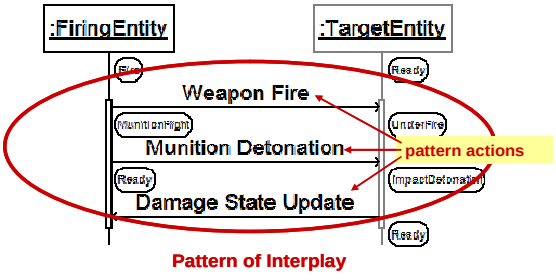

Figure 4 – Weapons Effect Pattern

3.2 Object Model Definition

The Object Model Definition section of the BOM Specification, with few exceptions, follows the formats prescribed by the HLA OMT. For the BOM, what was needed at the object modeling level was a way to describe data structures – specifically data types, object classes and the types of interactions that stakeholders need represented. HLA simply provides the most accepted and understood class structure mechanism for describing data types, object classes and interaction classes and that's why it is reflected by the BOM. The development group behind this standard didn't want to re-invent something that was already sufficient for M&S developers.

3.3 Model Mapping

The idea here is that the **abstract** things described in a Conceptual Model (entities and events) can be mapped to the actual types of things to be modeled and represented by a system implementation. Thus, if a firing entity is identified at the conceptual level (in the conceptual model), a Model Mapping indicates what object classes (or interaction classes) will fulfill the entities and events associated to it.

Incidentally it needs to be clearly understood that a BOM does not require within itself both Conceptual Model Definitions and Object Model Interfaces. Object Model interfaces can live /reside in other BOMs, or FOMs, or FOM Modules. In other words a Model Mapping can be made across one or more BOMs, FOMs or other architectures models (such as TENA) defining classes. This loose coupling capability is very important for facilitating composability and encouraging interoperability and reuse.

4. Evolution from BOMs to Modular FOMs

As discussed in Section 2 and 3, BOMs and Modular FOMs have similar characteristics, but some key differences based on the function they serve. These functions are best defined in terms of the HLA FEDEP, as illustrated in Figure 5. Step 2 of the FEDEP describes the need to develop an appropriate representation of the real world domain that corresponds to the immediate study or investigation. BOMs are especially well suited to this task, in that their composable structure allows for very rapid construction of conceptual models from small, reusable components. During Step 3, the functional capabilities defined by the aggregation of selected BOMs can be used to select federates, and also to define constraints on the federation design. It is also during Step 3 that emerging data exchange requirements are mapped to federate interfaces (i.e., who will publish and who will subscribe), and potentially applicable FOM subsets (i.e, Modules) are first identified. In Step 4, the FOM is fully developed, and decomposed into the specific FOM Modules needed to support the application. It is also at this time that the Model Mapping area of the BOM is used to associate elements of the conceptual model with one or more FOM Modules. In this way, traceability from conceptual model to design to implementation can be explicitly maintained.

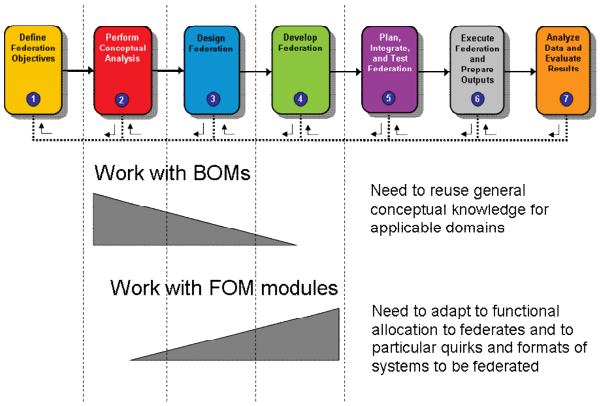

Figure 5: BOMs and FOM modules and FEDEP

Note that both BOMs and FOM Modules have strong reuse potential in this process. BOMs have a unique capability in capturing the conceptual model, and their atomic nature makes many BOMs applicable to multiple domains. The Model Mapping aspects of a BOM also help to identify potentially reusable FOM Modules, which themselves can also achieve broad reuse if specific federate implementations support (or can be converted to support) these particular data formats. This is illustrated in Figure 6.

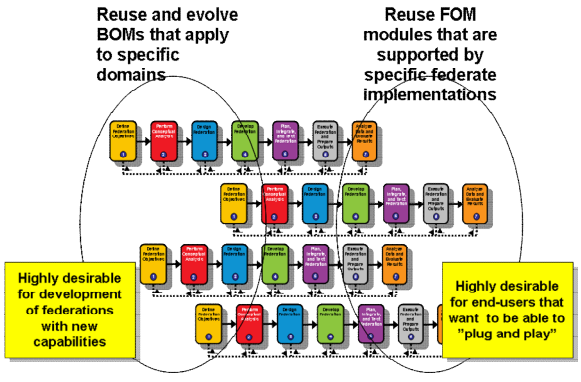

Figure 6: BOM/FOM Module Reuse

It is important to note that BOMs are not as tightly bound to HLA as a FOM or FOM module. FOM and FOM modules are HLA dependent, and anticipate the use of an RTI. BOMs, on the other had, while they may contain much of the same object model

information, are not as dependent on HLA, caring little about routing spaces, dimensions, MOM or the RTI.

4.1 Relationship between BOMs and FOM Modules

The questions that might be asked are as follows:

1. Can a BOM be a Modular FOM?
2. Can a Modular FOM be a BOM?
3. Can I use a BOM to define my Conceptual Model, and then map to Modular FOMs?
4. If I create a BOM Assembly of my BOMs, which have mappings to Modular FOMs, can it produce a viable FOM?

The answer to the first question is yes. While a BOM doesn't prescribe the use of all the HLA object model pieces it can still reflect these pieces. This is because the BOM imports the same OMT schema that a FOM does. Therefore, if a developer of a BOM chooses to include things such as dimensions in a BOM, they could do so, although this would limit the BOMs reuse potential for non-HLA implementations.

The answer to the second question is also yes. But keep in mind a BOM born from a FOM module would be limited. That is it would not have a conceptual model or model mapping elements and it would be hla dependent. Additionally, out of the box, it would not conform to the BOM Schema. Fortunately the model identification and object model interface (object classes, interaction classes, data types), would easily transfer to a BOM.

The answer to the third question is an emphatic **yes**. In fact, it is the position of the authors of this paper that BOMs be used to define the conceptual model with mappings formulated to FOM modules. Such an approach would help to facilitate the use of both BOMs and FOM modules. Consider that BOMs provide a clear understanding of intent and behavior at the conceptual model with mappings to specific models (i.e., FOM modules) that can be used in the HLA space. This combination has much merit.

The answer to the fourth question is maybe. In fact, the desire is that a BOM Assembly of BOM that has mappings to Modular FOMs would produce a viable FOM each and every time. But the truth is that there are elements of a FOM that are needed to work in the HLA space. If a FOM module neglects some of the required pieces for it function as a FOM, then it will need those pieces from another FOM module. An example would be the MOM module. If there is no MOM, then the aggregated set of FOM modules may not function properly in the HLA space with a given RTI.

5. Creating FOM modules from BOMs

BOMs can provide the "basis" for formulating FOM modules as well as complete FOMs. BOMs are attractive, because they can be created and defined with little attention made to some of the federation dependent details that a FOM would need. In other words as a developer, we can quickly craft what we need both at the conceptual model level and the object model level, with out concern for fine details that a FOM would need. This includes not only not having to define things such as dimensions and routing spaces, which may be federation dependent, but also not having to worry about the details of an object class and its attributes or interaction class and its parameters that might also be federate or federation dependent. Specifically, when an object class is defined with its attributes, the BOM cares very little about how Update Types, Update Conditions, Ownership, Publish / Subscribe role, available dimensions, transportation, or the order of such attributes are defined. Although they can be specified in a BOM, those are HLA implementation concerns at the federate and federation level, which, if assigned, limit the effective reuse value of a BOM.

When considering an iterative and incremental process to defining a FOM Module and FOMs, BOMs are an excellent approach. Additionally, the conceptual model perspective that a BOM brings is well needed within the M&S interoperability community, and should be highly encouraged. As object models are being formulated, it's a good first step to use the BOM as means to defining such object models. However, as those object models can be more specified for an HLA federate or federation, than they should be rolled into FOM modules. Mappings can still exist from a BOM connecting its conceptual model attributes with supporting FOM modules.

Additionally BOM Assemblies, representing an aggregate set of BOMs (and supporting FOM modules) can be used to generate complete FOMs. This is accomplished using mechanisms such as XSLT as depicted in Figure 7.

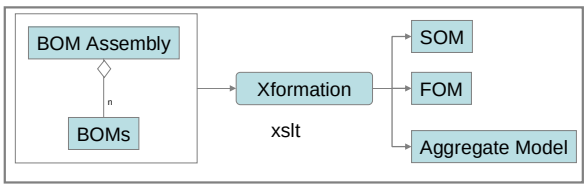

Figure 7 – How BOMs are Transformed

6. Summary

In this paper we have provided a brief overview of HLA Object Models (i.e., FOMs and SOMs), and Base

Object Models. Following our comparison of both BOMs and FOMs we explored how BOMs and FOM modules could be used collectively to facilitate reuse and help expedite development of HLA federates and federations.

The benefit of this approach is that it provides a highly effective component-like approach, which has been lacking within the M&S interoperability community. We conclude by recommending the use of both BOMs as a means to document the conceptual model aspects (patterns, states, entities, and events), and to formulate the basis for the HLA object models that can support such conceptual models.

We recommend also that FOM modules be developed based on these BOMs and serve as building blocks to help expedite the formulation of HLA federates and federations in a component-like way. Based on these capabilities, we also discourage the traditional approach of developing monolithic FOMs, which neglect the use of FOM modules and BOMs. Such an approach results in a FOM that is much more difficult to manage, maintain, and extend. BOM and FOM modules on the other hand, provide ample opportunities to (1) encourage interoperability across multiple architectures via the BOM, and (2) facilitate rapid deployment of customized HLA federations via the FOM modules).

References

[1] Nino Cocchiarella, “Conceptual Realism as Formal Ontology”, published in: Roberto Poli & Peter Simons (eds.) "Formal Ontology" - Kluwer Academic Publishers, Dordrecht/Boston/London 1996) pp. 27-60 - Nijhoff International Philosophy Series, vol. 53.

[2] IEEE 1516.4, "The HLA Federation Development and Execution Process (FEDEP)", www.ieee.org

[3] “The official MDA Guide Version 1.0.1”, OMG - the Object Management Group, http://www.omg.org/docs/omg/03-06-01.pdf

[4] XML Ref Bray, Tim; Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau (September 2006). “Extensible Markup Language (XML) 1.0 (Fourth Edition) - Origin and Goals” World Wide Web Consortium, http://www.w3.org/TR/2006/REC-xml-20060816/

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

[6] SISO, *BOM Template Specification*, SISO-STD-003-2006, SISO, 31 March 2006.

[7] 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. ref

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

[9] Björn Möller, Björn Löfstrand. ”Use Cases for the HLA Evolved Modular FOMs”, Proceedings of 2007 Euro Simulation Interoperability Workshop, 07E-SIW-040, Simulation Interoperability Standards Organization, June 2007

[10] Fowler, Martin (1996-11-27). “Analysis Patterns: Reusable Object Models.” Addison-Wesley. ISBN 0201895420.

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.

PAUL GUSTAVSON is chief scientist and co-founder of SimVentions, Inc. He has over 18 years experience supporting a wide variety of modeling and simulation, system engineering, and web technology efforts within the DOD and software development communities. Mr. Gustavson has been a long-time advocate and pioneer of the Base Object Model (BOM) concept for enabling simulation composability, interoperability, and reuse. He has also co-authored and edited several software development books and articles related to C++, UML, and mobile computing.

ROBERT LUTZ is a Principal Staff Scientist at The Johns Hopkins University Applied Physics Laboratory (JHU/APL). He has over 27 years of experience in the design, implementation, and evaluation of computer modeling and simulation (M&S) systems for military customers. He received his M.S. degree in Operations Research from the State University of New York at Stony Brook in 1980. Since joining JHU/APL in 1992, Mr. Lutz has assumed leadership roles on several M&S programs, including the Naval Simulation System

(NSS), Joint Warfare System (JWARS), and the Simulation Based Acquisition (SBA) initiative. Mr. Lutz also served as the technical editor for IEEE 1516.2 (HLA Object Model Template) and as working group chair for IEEE 1516.3 (HLA FEDEP). Currently, he is supporting a variety of M&S initiatives in support of the U.S. Modeling and Simulation Coordination Office (MSCO), the U.S. Joint Forces Command (USJFCOM), and the Navy's Broad Area Maritime Surveillance (BAMS) Program. He also serves as a guest lecturer in The Johns Hopkins University Whiting School of Engineering.

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.