Building and Using Base Object Models (BOMs) for Modeling and Simulation (M&S) focused Joint Training

Paul Gustavson
SimVentions, Incorporated
Fredericksburg, VA 22408
pgustavson@simventions.com

ABSTRACT

Modeling and Simulation (M&S) has proven to be an effective tool for training the warfighter and for planning and preparing against emerging threats within the global community. Key enablers that have contributed to this effectiveness have been the availability of models, tools, and other resources such as terrain databases, network assets, and scenario missions. And, as long as you remain within the mission area (i.e., Army, Navy) and domain (i.e., Surface, Air, Land) of your applied M&S framework (i.e., HLA, DIS, OOS), relative flexibility is achievable. The difficulty is in being able to leverage models across the environments, domains, and M&S frameworks to define and support scenarios and the executable environments representative of Joint training exercises. Models are almost exclusively defined for a particular simulation application. Fortunately, a standards-based approach, termed Base Object Models (BOMs), for describing and sharing models across environments, domains, and M&S frameworks has emerged from a simulation-community-wide grass-roots effort.

Essentially, BOMs can be used to represent the approaches and scenario elements that are necessary to fulfill specific military tasks, such as resupplying friendly forces or identifying and disarming combatants. Thus, they are well suited for supporting Joint training efforts. Furthermore, they serve not as executable models, but common descriptions of behavior, that can be implemented in multiple environments and frameworks.

This example-focused paper will largely center on building and using BOMs for composing Joint training environments. It will walk through the BOM development effort based on the Federation Development and Execution Process (FEDEP), and show how the Real-Time Platform Reference (RPR) BOMs and other supporting BOMs can be integrated and used to fulfill a specific training exercise. Additionally, this paper examines the language-neutral interface provided by the BOM and various implementation aspects that can be supported to enable federates.

ABOUT THE AUTHOR

PAUL GUSTAVSON is Chief Scientist and co-founder of SimVentions, Inc. He has over 16 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.

Building and Using Base Object Models (BOMs) for Modeling and Simulation (M&S) focused Joint Training

Paul Gustavson
SimVentions, Incorporated
Fredericksburg, VA 22408
pgustavson@simventions.com

INTRODUCTION

In the 21st century, the need for training among warfighters, mission planners, intelligence officers, and peacekeepers, has become paramount for the security and well-being of not only military and security personnel, but civilians here and abroad. We must plan and prepare against emerging threats within the global community. And to combat these threats, we must learn to defend and fight effectively and cooperatively in both Joint and combined environments. Lessons to be learned, however, should occur before our forces are deployed. An effective tool for preparing and training our individual, Joint, and coalition forces is the use of modeling and simulation (M&S). As explored in this paper, composability standards such as Base Object Models (BOMs) provide a viable mechanism to help establish effective M&S-based training environments.

Background

M&S is effective because it can be used to familiarize individuals in a safe environment for things that could eventually be played out in a hostile environment. Such training, through the use of simulation, can be done in a distributed manner without the expense of fuel, equipment, or need to move personnel. Furthermore, a major benefit of M&S applied within a distributed environment is that it allows for effective training, testing, and analysis.

The specific need for Joint training has become a principal focus within today's military. According to Navy Adm. Edmund P. Giambastiani Jr., who serves as North Atlantic Treaty Organization's (NATO's) Supreme Allied Commander Transformation (SACT) and the commander, U. S. Joint Forces Command (CDRUSJFCOM), emphasizes that, "[our effort is to try] to make our forces more integrated, more coherently integrated, so they can operate across a broad range of mission sets: peacekeeping, peacemaking, contingency operations, peace support, major combat operations, small-scale contingencies -- you name it (Sample, 2005)."

The use of M&S to support this Joint training need is being widely recognized. For example, U.S. Marine Capt. Erik Jilson, an M&S analyst at Quantico, Virginia, shares that, "Marine units must train to operate seamlessly in Joint and combined environments, but live Joint or combined training exercises are not always feasible." He adds that, "the [simulation] training that takes place before live training has the goal of better preparing Marines (Fisher, 2005)."

One application of distributed simulation used for warfighter training is illustrated in Figure 1 using the game Close Combat.

Figure 1 – Simulations Provide a Feel for Real Combat

Supporting the Need

Using M&S to establish virtual environments for training is proving to be very effective. Maj. John Basso, 1st Squadron, 10th Calvary Regiment, praised the advantages that working in virtual reality provided his men, "We can train the crews together. The skills we're learning [within the virtual environment] directly relate to Operation Iraqi Freedom (Churchill, 2005)."

Additionally, Tom Buscemi, director of IMEF's Battle Simulation Center Tactical Warfare Simulation Evaluation and Analysis System at Las Flores, Camp Pendleton, California, shares the following account, "A

senior watch officer who was in Iraq told me that with the exception of the causalities being real, what they experienced in Iraq was very similar to what the simulation produced (Fisher, 2005)." He added, "[This] emphasizes the effectiveness of the computer-driven combat simulations."

So, while computer-driven virtual combat simulations can be very effective for training the warfighter, the reality is that such environments can be difficult and arduous to build and establish. Yet, the increasing need, as we move forward, is to be able to compose and establish individual, Joint, and coalition focused training environments more rapidly and efficiently. As we will explore, the emergence of BOMs provides a viable mechanism to support this composability need.

M&S ENABLERS AND INHIBITORS

Known enablers that contribute to the effectiveness of M&S for training and other purposes such as testing and prototyping centers upon the availability of models, tools, and other resources such as terrain databases, network assets, and scenario missions. The availability of these resources, however, has evolved slowly relative to the emergence of other technologies such as computational devices, interfaces, displays, and network hardware available to consumers. For instance the availability of simulation software models is limited, and typically confined to a specific simulation and/or specific organization. Models representing the behavior characteristics of tanks, planes, and other platforms and subsystems are not typically developed for wide-spread use. The exception however is with 3D models, which provide the visual information for representing these platforms and entities. These visual models that have been developed often adhere to a standard format, and are often made available for purchase or download thereby enabling visual systems to provide realistic representation. In the same manner, simulation software models could also be developed for reuse and benefit the M&S community – in particular the Joint training community.

Essentially M&S tools are often limited by either a lack of availability to the community (i.e., limited distribution) or a lack of compliance to a set of adopted community standards (i.e., they are often limited to custom and/or proprietary solutions). Thus, the cross-use of M&S, despite the development of community standards including distributed technology standards such as Distributed Interactive Simulation (DIS) and the High Level Architecture (HLA), is often restricted to specific environments (i.e., Army, Navy, Air Force), domains (i.e., Land, Surface, Air) of interest – and sometimes specific programs within these environments and domains. Certainly within these environments, domains and programs, relative flexibility regarding the use of an M&S framework is achievable, but the difficulty, however, is in being able to leverage models across Joint environments, domains, and M&S frameworks. Such models (and tools) must be cross-leveraged and integrated to properly support the scenarios and the executable environments required for Joint training.

What must be encouraged is not only the consensus development of standards, but the consensus adoption of these standards as technology enablers for models, tools, and environments. And, if the integration of these models, tools, and environments is to occur, then these elements need to adhere to standard interfaces allowing connection and communication in a loosely coupled manner. Such capability complies with the concepts of a Service-Oriented Architecture (SOA), which should be a desired goal for any Joint training environment (Gustavson, Chase, Root, & Crosson, 2005).

Therefore, what is needed is the development of common models, services and databases that adhere to commonly understood standards, which can then be leveraged and used by complying tools and environments. The result of this would support one of the key capabilities required for the future of Joint training, which is the capability to integrate various models and simulations together quickly and easily, resulting in an execution environment that can support specific scenarios.

COMPOSABILITY DESIRE

Models, in general, are exclusively defined for a particular simulation application, and yet the need among all the stakeholders, which includes sponsors, designers, developers, testers, and users, is to be able to compose or put together exercisable environments, for test and play. The biggest desire, independent of what role we may support as stakeholders, is to compose things rapidly and efficiently. This is especially true when it comes to defining a Joint training environment.

Composability is defined as "The capability to select and assemble components in various combinations into complete, validated simulation environments to satisfy specific user requirements (Petty & Weisel, 2003)." This involves the selection of meaningful components, and the ability to couple these components together to

achieve the desired objectives for training (or perhaps other purposes such as testing or prototyping).

Some of the common barriers to achieving Composability are identified in Table 1.

Table 1 – Common Barriers of Composability

Barriers	Attitude Indicators of these Barriers
Time Constraints	• I really don't have much time. • I need a way to rapidly create a federation efficiently and effectively.
Ad-Hoc Development Temptation	• I've got the tools, I think I know what the customer wants, let's just jump in and start coding and integrating.
Lack of Conceptual Analysis	• What is conceptual analysis? I don't see the importance? • Oh, that's UML? We don't do UML.. • Why should I spend precious time on the conceptual analysis when I could be coding? • If I am going to do conceptual analysis, how do I integrate/use UML without buying expensive tools?
Maintaining Communication among Stakeholders	• The customer says he'll know what he wants when he sees it. • Let's get it done first then show the customer. • Let's not worry about the test guys.
Avoiding Not-Invented-Here (NIH) Syndrome	• If we didn't think of it, it can't possibly be right. • How can I create a model that benefits others? • Why should I create a model that others could benefit from? • I don't have time to focus on building to a new or different standard. • The standard doesn't meet our needs.
Discovering Reusable Components	• What exists out there already that I can leverage and reuse? • How do I discover / search for it? • How do I use the metadata? • What tools are out there to help in the discovery process?
Overcoming Proprietary Lock-In	• I need solutions that are not proprietary where I am not dependent upon a specific vendor's tools or component suite. • We've already invested in this solution.
Dealing with Complex Integrations	• How can I easily plug my simulation into the world of HLA?
Supporting Multiple Federations	• I can only support one Federation at a time.
Manageability Issues	• Hmm, I've created / inherited this big, monolithic, unmanageable Federation Object Model (FOM) developed for my federation. • How can all my people work on it collaboratively? • How can I add extensions without breaking it? • How can I better configuration manage (CM) it?

Composability Process

The enablers to achieving composability and overcoming these barriers center upon adherence to a process, and that process must include the interest of all the stakeholders. One process commonly described and used within the M&S community is the Federation Development and Execution Process (FEDEP). As illustrated in Figure 2, there are two aspects of composability that a process like FEDEP encourages: Model Composability and System Composability.

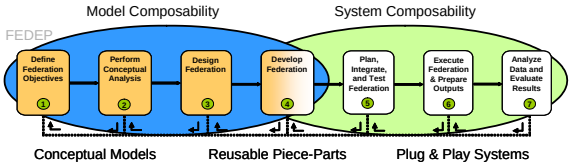

Figure 2 – Composability Process as it Relates to the FEDEP

Often the focus is on the right-hand side: System Composability. Here, the desire is to achieve plug and play systems. While System Composability is a worthy goal, there is an increased chance of success if time is spent on first achieving Model Composability, which is often neglected.

Activities

The activities involved in achieving Model Composability begin with the collection of requirements and identifying what is needed for the simulation environment as defined by FEDEP Step 1. This is followed by performing a conceptual analysis as defined by FEDEP Step 2. The effect of this analysis will result in a collection of conceptual models ordered hierarchically in terms of interest and detail. From the FEDEP perspective, the conceptual model identifies "what the [simulation or federation] will represent, the assumptions limiting those representations, and other capabilities needed to satisfy the user's requirements (IEEE 1516.3, 2003)."

The types of things to identify from these conceptual models are patterns – or common sets of recurring behavior that occur in accomplishing a common objective, capability, or purpose. The recognition of patterns is proving to be a key approach for supporting system design and software development, yielding a framework for components, which is identified as reusable piece-parts in the FEDEP view. These reusable piece-parts can be used to support model composability.

Certainly the means for enabling composability, including support for analysis, development of models, and reuse of components (i.e., reusable piece-parts) are accomplished through the use of tools. Tools are paramount. However, even with tools in hand, it is also important to have the availability of models via a set of libraries that these tools can use. These libraries

are ones that should be populated and updated by community members who may also be developing and composing simulations and simulation environments.

Common Standard for Simulation Components

However, the key to offering reusable piece-parts and tools that help support the job of model composability is that the models representing these reusable piece-parts need to be based on a common format and standard. Equally, the tools need to be able to leverage that standard. One such approach that has been developed for M&S to support composability is the Base Object Model (BOM), which provides a framework for component standards.

The BOM is defined as, "a piece part of a conceptual model composed of a group of interrelated elements, which can be used as a building block in the development and extension of simulations and simulation environments (SISO-STD-003.1-DRAFT-V0.11, 2005)."

BOMs can serve the needs of the M&S community, providing the framework from which to define common interfaces for integrating models and the framework from which to define meaningful content that can be shared and used among Joint players.

However, the key for unlocking the BOM capability to support Model Composability is to first understand how the conceptual model can be defined, used, and shared among all stakeholders.

FOCUSING ON THE CONCEPTUAL MODEL

The first step in carrying out any type of development task is to understand what needs to be represented. For successful model composability, understanding what needs to be represented is paramount; especially in regards to using BOMs to establish environments for purposes such as Joint training. Without understanding what needs to be represented, the effectiveness of the environment in facilitating training at any level is compromised. This is why the conceptual model is important. Essentially, the conceptual model can be a useful communication mechanism across each stage of the development process, encouraging collaboration among stakeholders and modularity of design.

Ideally there should be multiple conceptual models identified for describing the environment to be represented. This modularity provides the basis for defining reusable models and components, and, subsequently, permits better development among team members, improving configuration management, and simplifying the burden of unit testing. Modularity is achieved by breaking the problem domain down into parts that can be addressed separately. This results in a manageable collection of conceptual models. Furthermore, the information identified for a conceptual model can be used to find reusable components or as the basis for creating new ones.

Common Aspects of a Conceptual Model

It is important, however, to understand what these conceptual models contain; that is what should be captured and reflected within a conceptual model. Some of the discernable attributes of a conceptual model are defined as follows (Gustavson, Zimmerman, & Turrell, 2003):

- Describes functional and behavioral capabilities
- Maps to objectives / stakeholder requirements
- Identifies conceptual entities to be represented
- Identifies logic and algorithms
- Identifies relationships
- Identifies assumptions and limitations

Based on these attributes, it is clear that the goal is to identify "what" needs to be represented, and, at a high level, the activities and relationships that take place. This might include, for example, the common scenario elements that are necessary to fulfill specific tasks such as resupplying friendly forces or identifying and disarming combatants.

This set of attributes correlates with the pattern concept described earlier, which focuses on identifying common sets of recurring behavior that occur in accomplishing a common objective, capability, or purpose. And the BOM provides a template for capturing the end-state of such a conceptual model.

As described in the BOM Template Specification, the aspects of a simulation conceptual model that are contained in a BOM include static descriptions of items resident in the real world, which are described in terms of conceptual entities and conceptual events, and contain information on how such items relate or interact with each other in the real world in terms of Patterns of Interplay and state machines. This relationship among these conceptual model elements supported by the BOM is depicted in Figure 3.

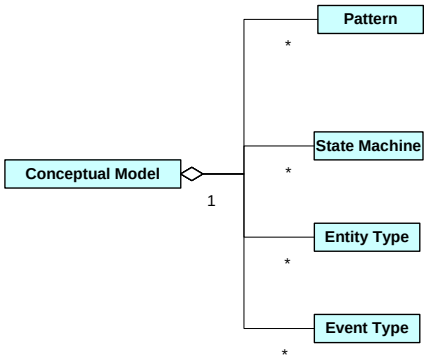

Figure 3 – Conceptual Model Elements of a BOM

These elements of a conceptual model, which can be described using a BOM, are useful when the simulation software designers begin to consider what their simulation will need to do.

Component Interfaces and the Support for SOA

The BOM also provides a mechanism to define the required simulated capabilities reflected in the conceptual model in the context of an interface description. This interface is described in terms of object-based classes defining not only the capabilities of a component but the simulation application that uses it. What is significant about this type of interface is that it encourages a service-oriented architecture (SOA) approach.

SOA is an architectural approach focused on the interoperability and loose coupling of integrated elements (Gustavson, Chase, Root, & Crosson, 2005). This could include the mix of live, virtual, and constructive simulations patched in together, and, although each element is distinct and disparate, they are capable of interoperating seamlessly. In fact, the chief objective of SOA is to minimize unnecessary dependencies among systems and software elements (i.e., components) such that different implementations, which could potentially be manufactured by different vendors, can integrate together to provide some repeatable and reliable service and capability. This objective should be the same objective required for creating Joint training environments. We need to be able to patch-in models, from the model composability standpoint, as well as live, virtual, and constructive simulations, from the system composability standpoint, to create effective environments for Joint training across the various forces and organizations.

What BOMs provide is a mechanism for supporting the creation of "common interfaces," which is the "key tenant for supporting a loose coupled simulation environment (Gustavson, Chase, Root, & Crosson, 2005)." From an SOA perspective, this common interface that BOMs provide allows federates to act as software agents, playing either the role of producer or consumer establishing a composable environment for Joint training.

Figure 4 illustrates how BOMs can be coupled together to formulate such a composable environment.

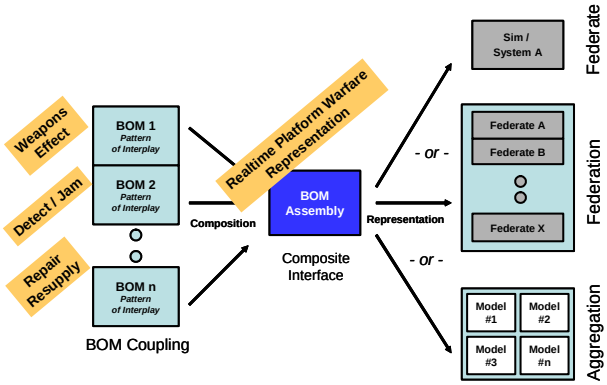

Figure 4 – Composition of BOMs

In the next section we will explore how a BOM is built and how a collection of these BOMs can be coupled together for establishing a federate, federation, or an aspect of a federation being represented as an aggregate.

DEVELOPING BOMS AND BOM COMPONENT IMPLEMENTATIONS

As we have already explored in the last section, the BOM is made of several key elements. These elements are illustrated in Figure 5.

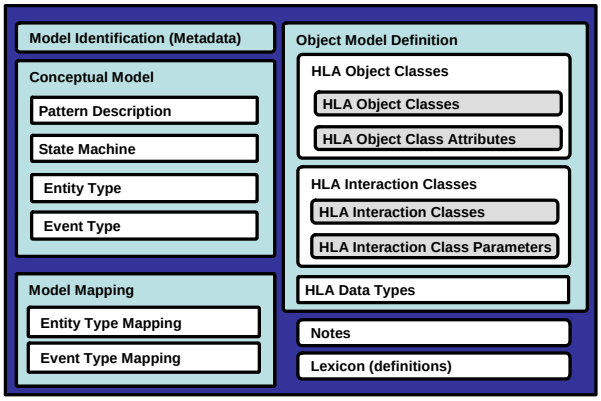

Figure 5 – BOM Elements

In this section we will explore what information can be captured within each of these BOM template elements. The example we will look at is based on a common pattern found within most military-centric simulation exercises, and that is a weapons effect pattern. We will then show how to take a collection of different models supporting unique conceptual models and combine them into a BOM Assembly, which can be used to define a Joint training environment.

Core and Common Metadata

The one element that we haven't talked about to any extent is the Model Identification element, and yet this may be the most important. The Model Identification provides the means to identify and tag the critical metadata for cataloging conceptual models and components. Metadata can be simply described as data about data. It is a way to label and describe information and is used "to aid in the identification, discovery, assessment, and management" of that information ("Final Report on Metadata," 2000)."

The Model Identification provides a structure to document what a BOM (or even other models) is all about – what it contains. One would not purchase a bottle of medicine without first seeing and understanding the label. An unlabeled medicine bottle, soda can, or packaged food product, would go unused. Its content never understood. Likewise, it is important to label a conceptual model or component. The Model Identification provides the labeling information as illustrated in Figure 6.

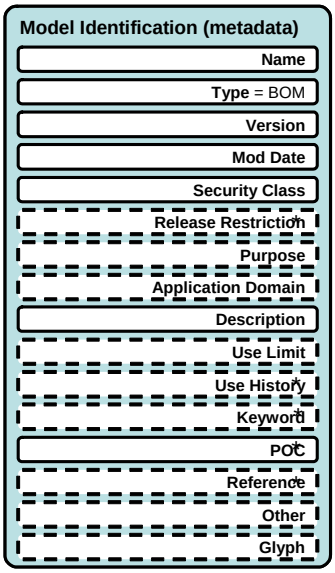

Figure 6 – Model Identification Metadata Elements

This Model Identification structure is based on a combination of other metadata efforts and products such as Dublin Core, the Department of Defense (DoD) Discovery Metadata Specification (DDMS), VV&A Recommended Practice Guide (RPG), and the HLA Object Model Template (OMT), IEEE 1516.2.

The following suggestions are recommended as pointers to filling out the Model Identification for a conceptual model or component which is captured within a BOM, (Gustavson, Scrudder, Lutz, & Bachman, 2005).

- First, document a model in such a way that its *Purpose* is clear, *Use Limitations* are identified, the *Application Domain* is understood, and multiple *POCs*, such as sponsor representative, and developers are known.
- Encourage integration experience of models to be fed back into the *Use History*. This is vital for increasing use of the model to support various efforts such as creating a Joint training exercise environment.
- Use the mechanisms provided to help manage and control the model through the *Version*, *Security Class*, and *Release Restriction*.
- Take advantage of the ability to *Reference* other information such as supporting documents, databases, scenarios, and 3D models.
- In general, keep things descriptive yet concise. This allows candidate models to be more readily found and reused. And, just because a field may be optional doesn't mean necessarily that it should be ignored.

Table 2 provides an example of a completed Model Identification for labeling a weapons effect model, which is a BOM that could be used in a Joint training environment.

Table 2 – Model Identification Example

Category	Information
Name	WeaponsEffect
Type	BOM
Version	1.0
Modification Date	2004-11-19
Security Classification	Unclassified
Release Restriction	Not for release outside the I/ITSEC community
Purpose	RPR FOM decomposition
Application Domain	Realtime Platform Simulation
Description	This is an example BOM
Use Limitations	None
Use History	Initial release
Keyword	
Taxonomy	Military Warfare
Keyword Value	Engagement

POC	
POC Type	Primary author
POC Name	P. Gustavson
POC Organization	SimVentions
POC Telephone	540 372-7727
Reference	
Ref Type	Glossary
Identification	ISBN 12345678901
Reference	
Ref Type	Conceptual Model
Identification	http://boms.info/
Other	na
Glyph	
Type	jpg
Alt	WeaponEffectPicture1
Height	32
Width	32

Filling in the Conceptual Model

Now that we know how to label a BOM, which we might want to use for supporting the composition of a Joint training environment, it's important to fill the BOM with content so that it can be effective. We start with filling in the conceptual model so that we can fully understand its application.

Earlier we talked about the aspects of a conceptual model and that the BOM includes elements for capturing static descriptions in terms of conceptual entities and events, and the relationship of those elements in terms of Patterns of Interplay and state machines. The essence of the conceptual model for our example is depicted in the Figure 7.

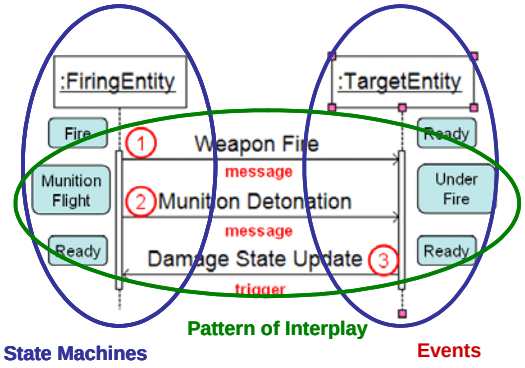

Figure 7 – Weapons Effect Conceptual Model View

This sequence diagram defined using the Unified Modeling Language (UML) illustrates the Pattern of Interplay for our Weapons Effect example model. A Pattern of Interplay refers to the sequence of activities related to one or more conceptual entities. Our conceptual entities include a Firing Entity and a Target Entity. Identified in this view, are the states associated with each conceptual entity. The Events that must support the interplay are also represented.

This provides a simple view of our model, but it still must be captured in a convention that allows it to be integrated and reused. The guidance for filling the Patterns of Interplay, State Machines, Entity Types, and Event Types is provided in the "Guide for BOM Use and Development," which has been developed as a support document for the "BOM Template Specification." However, examples for each of these tables are provided below.

Table 3–Pattern of Interplay for Weapons Effect

Action	
Sequence	1
Name	WeaponFireAction
Event	WeaponFire
Sender	FiringEntity
Receiver	TargetEntity
BOM	NA
Action	
Sequence	2
Name	MunitionDetonationAction
Event	MunitionDetonation
Sender	FiringEntity
Receiver	TargetEntity
BOM	NA
Action	
Sequence	3
Name	DamageStateUpdateAction
Event	NA
Sender	TargetEntity
Receiver	FiringEntity
BOM	DamageStateUpdateBOM

Table 4 – State Machine Example

Name	FiringEntity	TargetEntity
Conceptual Entities	Lifeform	Lifeform
	Platform	Platform
State		
Name	Ready	Ready
ExitAction	CommandToFire	WeaponFireAction
NextState	Fire	UnderFire
Notes		
State		
Name	Fire	UnderFire
ExitAction	WeaponFireAction	MunitionDetonationAction
NextState	MunitionFlight	MunitionFlight
Notes		
State		
Name	MunitionFlight	ImpactDenotation
ExitAction	MunitionDetonationAction	DamageStateUpdateAction
NextState	Ready	Ready
Notes		

Table 5 – Entity Types Example

Entity Type			
Name	Description	Characteristics	Description
FiringEntity	Thing that fires a weapon at a target	ID	Unique id for entity
		Location	Physical position of the entity
TargetEntity	Thing that is the intended target of a weapon fire	ID	Unique id for entity
		Location	Physical position of the entity
		Velocity	Velocity for the entity

It should be noted that the FiringEntity and TargetEntity could be supported by the same EntityType. However, we have separated it out for our example.

Table 6 – Event Type Table Example

Event Type			
Name	Description	Characteristics	Role
WeaponFire	Message for representing weapon fire interaction among conceptual entities	FiringEntity_identifer	Source
		TargetEntity_identifer	Target
		Munition_identifer	Content
MunitionDetonation	Message for representing weapon detonation interaction among conceptual entities	FiringEntity_identifer	Source
		TargetEntity_identifer	Target

The question that is often wondered, but perhaps seldom asked, is how all four of these conceptual model elements tie together. The best way to understand the relation of these four elements is to study the illustration in Figure 8.

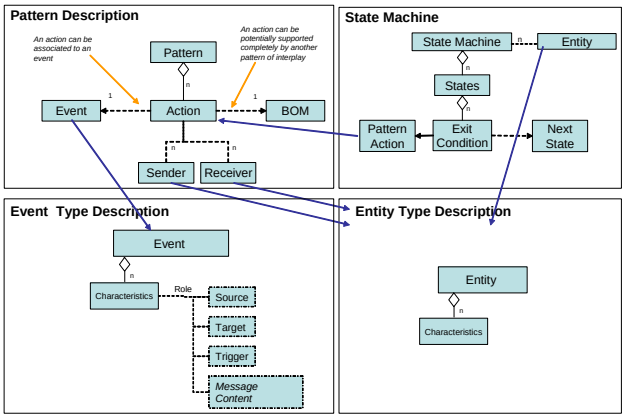

Figure 8 –BOM Conceptual Model Relationship

Defining the Interface Element (aka Object Model)

Another aspect of BOM content that is useful is the interface description. For a BOM, this interface description, which is identified as the Object Model Definition, is described using HLA Object Model Template (OMT) constructs. The specific HLA OMT constructs used include: HLA object classes, HLA interaction classes, and their attributes and parameters. The use of HLA OMT provides a familiar construct for the simulation software designer, however it is not intended to restrict the use of a BOM to HLA specific implementations.

The guidance for filling in the HLA object classes, HLA interaction classes, attributes and parameters for use within a BOM is discussed in the "Guide for BOM Use and Development," which also refers to the HLA OMT Specification for understanding these tables (SISO-STD-003.0-DRAFT-V0.11, 2005). Examples for each of these OMT-based tables used to represent a BOM are provided below.

Table 7 – HLA Object Class Table Example

HLA Object Classes				
HLAobjectRoot	BaseEntity	PhysicalEntity	Platform	
			Lifeform	Human

Table 8 – HLA Interaction Class Table Example

HLA Interaction Classes	
HLAinteractionRoot	WeaponFire
	MunitionDetonation

Table 9 – HLA Attributes Table Example

HLA Attributes									
Object	Attribute	Datatype	UpdateType	UpdateCondition	Ownership	P/S	AvailableDimensions	Transportation	Order
HLAobjectRoot	HLAprivilegeToDelete	NA	NA	NA	NA	NA	NA	HLAreliable	Receive
BaseEntity	Spatial	SpatialStruct	Conditional	NA	No Transfer	PS	NA	HLAreliable	Receive
PhysicalEntity	ForceIdentifier	ForceIdentifierEnum8	Conditional	OnChange	No Transfer	PS	NA	HLAreliable	Receive
				.					
Platform	HatchState	HatchStateEnum32	Conditional	OnChange	No Transfer	PS	NA	HLAreliable	Receive
Lifeform	StanceCode	StanceCodeEnum32	Conditional	OnChange	No Transfer	PS	NA	HLAreliable	Receive

Table 10 – HLA Parameters Table Example

HLA Parameters		
Interaction	Parameter	Datatype
MunitionDetonation	MunitionObjectIdentifier	NA
MunitionDetonation	RateOfFire	SpatialStruct
MuntionDetonation	MunitionType	ForceIdentifierEnum8
WeaponFire	TargetObjectIdentifier	HatchStateEnum32
WeaponFire	WarheadType	StanceCodeEnum32

Defining a common interface using HLA OMT constructs, which provides a familiar syntax to simulation engineers, allows it to be implemented in multiple environments and frameworks.

Conceptual Model and Object Model Mapping

Now that we have filled our BOM with both Conceptual Model content and Object Model content, we should also provide a mapping between the Conceptual Model view and the interface elements of the Object Model view, otherwise it is difficult to understand how the conceptual model can be fulfilled and what the intended capabilities of the object model are. This provides the basis for simulation software design and for the interchange among other simulations, which is particularly important for defining a Joint environment for training.

BOMs provide a mechanism for mapping this relationship through an Entity Type Mapping and an Event Type Mapping and are illustrated in the next two tables.

Table 11 – Entity Type Mapping Example

Entity Type Mapping Table				
Entity Type	HLA Object/Interaction Class	Characteristics	HLA Attributes/Parameters	Condition
FiringEntity	HLAobjectClassRoot.BaseEntity.PhysicalEntity.Lifeform.Human	ID	Human.EntityIdentifier	na
		Location	Human.Spatial.SpatialFP.WorldLocation	na
TargetEntity	HLAobjectClassRoot.BaseEntity.PhysicalEntity.Platform	ID	Platform.EntityIdentifier	na
		Location	Platform.Spatial.SpatialFP.WorldLocation	na
		Velocity	Platform.Spatial.SpatialFP.VelocityVector	na

Table 12 – Event Type Mapping

Event Type Mapping Table				
Event Type	HLA Object/Interaction Classes	Characteristics	HLA Attributes/Parameters	Condition
WeaponFire	HLAinteractionRoot.WeaponFire	FiringEntity_Identifier	HLAinteractionRoot.WeaponFire.FiringObjectIdentifier	na
		TargetEntity_Identifier	HLAinteractionRoot.WeaponFire.TargetObjectIdentifier	na
		MunitionEntity_Identifier	HLAinteractionRoot.WeaponFire.MunitionObjectIdentifier	na
MunitionDetonation	HLAinteractionRoot.MunitionDetonation	FiringEntity_Identifier	HLAinteractionRoot.MunitionDetonation.FiringObjectIdentifier	na
		TargetEntity_Identifier	HLAinteractionRoot.MunitionDetonation.TargetObjectIdentifier	na

SUMMARY

We have now completed filling in the content that can be contained in a BOM. This type of fully loaded BOM spanning both conceptual model and the interface elements offered by object model, and those BOMs that may be heavy on either the interface side or conceptual side can be selected, connected, and coupled together to formulate a BOM Assembly. The process for this is illustrated in Figure 9.

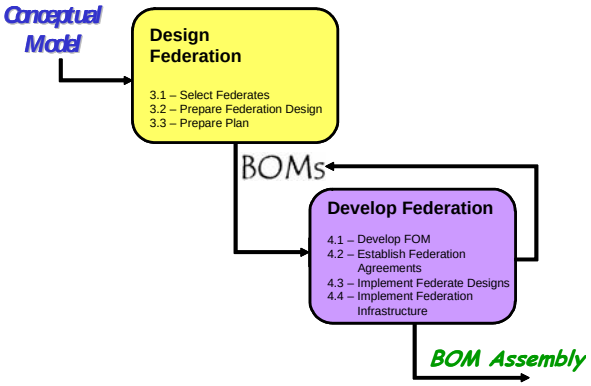

Figure 9 – BOM Integration Activities

From our example, what we might be able to compose is illustrated in Figure 10.

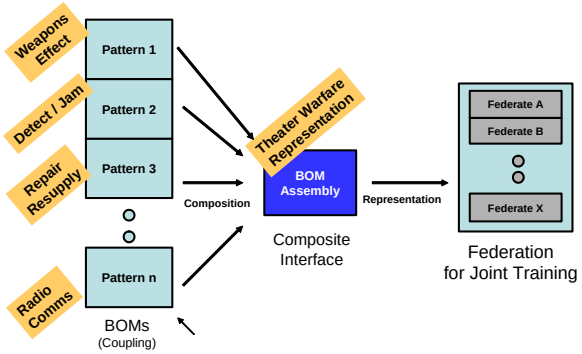

Figure 10 – BOM Composition

BOMs and BOM Assemblies represent the capabilities and interface required for defining exchange in a loosely coupled Joint training environment thereby supporting both Model Composability and System Composability as illustrated in Figure 11.

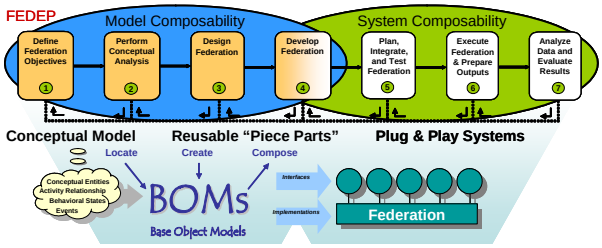

Figure 11 – Application of BOMs in the Composability Process

Individually, BOMs also allow us to define separate BOM component implementations (BCIs) that can cater to a specific computational platform, operating system, or language. So that while the implementation is unique, the interface is common, and therefore different BCIs could be produced, even for different resolution needs and yet still support the same BOM interface and capabilities described by its conceptual model.

Ultimately, BOMs provide utility in not only being able to capture training module elements in the context of simulation, but also in combining those module elements so that they can provide an effective environment for training and testing.

Truman C. Preston, Assistant Chief of Staff, G7, II Marine Expeditionary Force, Marine Corps Base Camp Lejeune, N.C. sums it up in saying that "Simulations are more cost effective ways to build in the repetitions needed to gain experience (Fisher, 2005)." The key is offering the right simulation to those that need it, and in this era that need spans the Joint community. Therefore, BOMs are intended to provide a mechanism for identifying capability via the metadata, to describe that capability via the conceptual model, and finally to define the content for modeling that capability via the object model definition. Additionally, BOMs provide a mechanism for mapping between the conceptual and the interface elements of an object model.

Consider that one of the principal difficulties in establishing environments for training can be in locating and leveraging models across environments, domains, and M&S frameworks. However, we have explored how BOMs provide a viable standards-based mechanism useful in supporting the composability of simulation and simulation environments, and conclude that BOMs are well suited for supporting the needs of the Joint training community.

ACKNOWLEDGMENTS

Each of the following individuals has provided significant help in the development of the BOM concept that has been described. The author wishes to thank them for their countless contributions and steadfast support.

- Björn Löfstrand (Pitch)
- Bob Lutz (JHU APL)
- Chris Rouget (Preforce)
- Chris Turrell (Alion/DMSO)
- Jake Borah (Aegis Research)
- Jane Bachman (NSWCDD - TEAMS)
- Larry Root (SimVentions)
- Mark Biwer (Northrop Grumman)
- Reed Little (CMU SEI)
- Roy Scrudder (ARL/UT)
- Steve Goss (SimVentions)
- Steve Reichenthal (Boeing)
- Tram Chase (SimVentions)

In addition, the author would like to thank the Defense Modeling and Simulation Office (DMSO) for sponsoring the BOM development effort intended to bring forth an increase in composability within the distributed simulation communities, and also the U.S. Army, specifically RDECOM, and PEOSTRI, which have sponsored a Small Business Innovative Research project associated with the application of BOMs for supporting aggregation and load balancing of distributed simulation environments.

REFERENCES

Task Force on Metadata, "The Final Report of the Association for Library Collections and Technical Services," (2000)

C. Fisher, "Simulations Add 'Playtime' To Training Cycle," Headquarters Marine Corps, Story Id#: 2005412101145, http://www.marines.mil/marinelink/mcn2000.nsf/lookupstoryref/2005412101145

Gustavson, Chase, Root, Crosson, "Moving Towards a Service-Oriented Architecture (SOA) for Distributed Component Simulation Environments," 05S-SIW-091

Gustavson, Scrudder, Lutz, Bachman, "Understanding the BOM Metadata and Making It Work For You," 05S-SIW-084

Gustavson, Zimmerman, Turrell, "Capturing Intent-of-Use for the Conceptual Model - A Key to Component Reuse," 03F-SIW-080

IEEE 1516.3, IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP), March 2003

M. D. Petty and E. W. Weisel, "A Composability Lexicon," *Proceedings of the Spring 2003 Simulation Interoperability Workshop*, Orlando FL, March 30-April 4 2003, 03S-SIW-023

Sgt. 1st Class Doug Sample, USA, "Facing the Future: Transforming DoD Is 'Constant Process'," American Forces Press Service, http://www.defenselink.mil/news/Mar2005/20050328_331.html

SISO, "Base Object Model (BOM) Template Specification," SISO-STD-003.1-DRAFT-V0.11

SISO, "Guide for Base Object Model (BOM) Use and Implementation," SISO-STD-003.0-DRAFT-V0.11

Spc. Allison Churchill, USA, "Soldiers Simulate Battlefield Conditions," 4th Infantry Division Public Affairs, http://www.defenselink.mil/transformation/articles/2005-03/ta030405d.html