Simulation Interoperability Standards Organization (SISO)

Guide for Base Object Model (BOM) Use and Implementation

SISO-STD-003.1-2006

31 March 2006

Prepared by:
SISO Base Object Model Product Development Group

Keywords: Automation, Behavior, BOM, Components, Composability, Conceptual Model, FEDEP, Interoperability, Metadata, Patterns, Requirements, Reuse

SISO Inc. Board of Directors
P.O. Box 781238
Orlando, FL 32878-1238, USA

Acknowledgements

This document was created as a community effort by the Base Object Model Product Development Group (PDG). This PDG was chartered by the Simulation Interoperability Standards Organization (SISO) Standards Activity Committee (SAC) on 15 January 2003. This document would not have been possible without the hard work and dedicated efforts of the following individuals:

Product Development Group Officers

Lawrence M. Root (Chair)
Bob Lutz (Vice-Chair)
Jane T. Bachman (Secretary)
Chris Rouget (SAC TAD)

Drafting Group Team

Paul Gustavson (Lead Editor)
Jake Borah
Tram Chase
Stephen M. Goss
Björn Löfstrand
Steven W. Reichenthal
Roy Scrudder
Chris Turrell

Additional Document Contributions

John Hancock
Bob Lutz
Chris Stapleton
George Hughes (Assigned Reviewer)
Staffan Löf (Assigned Reviewer)
Troy Sizemore (Technical Editing)

Balloting Group

Jane T. Bachman
Mark Biwer
Curtis Blias
Jack Borah
Don Brutzman
Ernst Budde
Robert Chapman
Tram Chase
Tim DiVecchia
Reginald Ford
William Gordon
Stephen M. Goss
Paul Gustavson
Michael Hieb
Nathalie Le Rest
Reed Little
Björn Löfstrand
Bob Lutz
Alastair MacDonald
Vahid Mojtahed
Katherine L. Morse
Steven Reichenthal
Chris Rouget
Mike Rutledge
Geoffrey Sauerborn
Randy Sanders
Roy Scrudder
Graham Shanks
Steven Sheasby
Susan Solick
Joe Sorroche
Eugene Stoudenmire
Andreas Tolk
Charles Turnista
Chris Turrell
Roger D. Wuerfel

This page left intentionally blank

Table of Contents

1 Introduction

The Simulation Interoperability Standards Organization (SISO) focuses on facilitating simulation interoperability across government and non-government applications worldwide. SISO's interests include methods that support and promote reuse of simulation components; agile, rapid, and efficient development and maintenance of models; as well as, integration of models into operational systems or embedding real-world systems into virtual environments.

Base Object Models (BOMs) provide a component framework for facilitating interoperability, reuse, and composability. 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. The interplay within a simulation or federation can be captured and characterized in the form of reusable patterns. These *patterns of interplay* are sequences of events between simulation elements. The representation of the pattern is captured in the BOM.

There are two BOM related documents developed by the SISO BOM Product Development Group (PDG). Current documents include the “Base Object Model (BOM) Template Specification” and the “Guide for Base Object Model (BOM) Use and Implementation.” This document is the “Guide for Base Object Model (BOM) Use and Implementation.”

1.1 Purpose

This “Guide for Base Object Model (BOM) Use and Implementation” introduces methodologies for creating BOMs and implementing them in the context of a larger simulation environment. The document is a means of familiarizing the reader with the concept of BOMs and providing guidance for BOM development, integration, and use in supporting simulation development.

1.2 Scope

The “Guide for Base Object Model (BOM) Use and Implementation” provides guidance and direction for BOM authors and users interested in creating and/or integrating BOMs in a variety of their manifestations.

BOMs serve to provide an end-state of a simulation conceptual model and can be used as a foundation for the design of executable software code and integration of interoperable simulations. The aspects of a simulation conceptual model found in a BOM contain static descriptions of items resident in the real world described in terms of *conceptual entities* and *conceptual events*. In addition, those aspects of a simulation conceptual model found in a BOM 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*. Both these static and dynamic views of a conceptual model, which can be described using a BOM, are useful when the simulation software designers begin to consider what their model, simulation, or federation will need to do.

The required simulated capabilities, which are reflected in the conceptual model, can also be defined in the context of an interface description that represents the information necessary for execution and exchange. This interface information is often described in terms of class structures that collectively define the inherit capabilities of a simulation application. For a BOM, this interface description is defined using the High Level Architecture (HLA) Object Model Template (OMT) constructs specifically in terms of *HLA object classes*, *HLA interaction classes*, and their *attributes* and *parameters*. The use of the HLA OMT provides a familiar construct for the simulation software designer, but does not restrict the use of a BOM to HLA specific implementations.

Also important in supporting simulation development, is to understand the relationship of a simulation conceptual model with the class structures of an object model, which may be used as the basis for simulation

software design and for the interchange among other federates. In this capacity, BOMs provide a construct for mapping the relationship between the entity and event elements of the Conceptual Model Definition and the class structure elements of the Object Model Definition, which are described using HLA OMT constructs.

In regards to these essential elements, this “Guide for Base Object Model (BOM) Use and Implementation” provides discussion on BOM development and the application and use of BOMs for the assembly of simulations and simulation spaces, which is illustrated in Figure 1-1.

This “Guide for Base Object Model (BOM) Use and Implementation” should be used in conjunction with the “Base Object Model (BOM) Template Specification” which provides the standard defining semantics and syntax needed to represent a BOM.

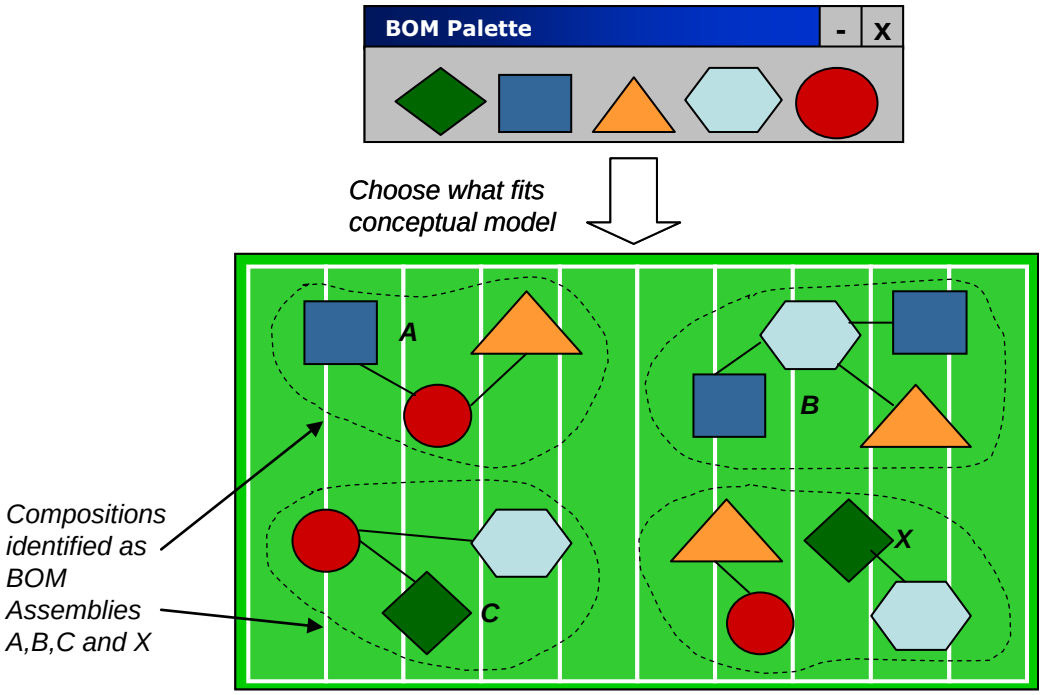

Figure 1-1 BOM Composability View

The large rectangular region in Figure 1-1 represents the simulation environment in which BOMs may be composed and used. The items marked A, B, C, and X each represent capabilities to be supported through the composition of BOMs. These compositions are known as BOM Assemblies. The lines depicted within an assembly represent the references that can be made between BOMs. As depicted in Assembly X, not all BOMs within an assembly require an explicit reference to other BOMs.

1.3 Objective

BOMs provide a mechanism for defining a simulation conceptual model and optionally mapping to the interface elements of a simulation or federation using HLA OMT constructs. The objective is to encourage reuse, support composability, and help enable rapid development of models, simulations, and federations. In support of this objective, this document provides guidance information for those developing and/or using BOMs.

1.4 Intended Audience

This document is intended for individuals and organizations in the Modeling and Simulation (M&S) community who are interested in the modeling, interoperability, reusability, componentization, and composition of models, simulations, and federations. Potential consumers of this specification include those involved in the military (both U.S. and International) that use virtual, constructive, and/or live simulations for the purpose of testing, training, analysis, or acquisition, and/or operational use. Also, those involved in the commercial sector including education, entertainment, manufacturing, medical, and other markets may find this specification useful in establishing interoperable environments.

This page left intentionally blank

2 References

Several specifications, documents, and technical references provide the technical foundation for designing and developing BOMs and BOM-based federates and federations. This guide should be used in conjunction with the publications listed in Tables 2-1 and 2-2. If any of the specifications identified in Tables 2-1 or 2-2 are superseded by an approved revision, then the revision shall apply.

Table 2-1 Primary Reference Documents

Document Number	Title
SISO-STD-003.1	Base Object Model (BOM) Template Specification
IEEE Std 1516.2-2000*	IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template (OMT) Specification
XML 1.0 (Second Edition)	Extensible Markup Language (XML) 1.0 (Second Edition) W3C Recommendation, 6 October 2000
XML Schema Part 1	XML Schema Part 1: Structures W3C Recommendation 02 May 2001
XML Schema Part 2	XML Schema Part 2: Datatypes W3C Recommendation 02 May 2001

Table 2-2 Other Reference Documents

Document Number	Title
IEEE Std 1516-2000*	IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules
IEEE Std 1516.1-2000*	IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification
IEEE Std 1516.3-2003	IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP)
IEEE SP1122 ISBN 0-7381-2601-2	Authoritative Dictionary of IEEE Standard Term for Reference
N/A	BOM PDG Product Nomination – March 2003
SISO-REF-005-2001	Base Object Model (BOM) Study Group Final Report – May 2001
SISO-REF-001-1998	Reference FOM (RFOM) Study Group Final Report - March 1998
W3C NOTE-SRML-20021218	SRML – Simulation Reference Markup Language W3C Note, http://www.w3.org/TR/SRML/, 18 Dec 2002
ANSI/NISO X39.85-2001	National Information Standards Organization, “The Dublin Core Metadata Element Set,” September 2001

DoD Discovery Metadata Standard	Deputy Assistant Secretary of Defense (Deputy Chief Information Officer), “Department of Defense Discovery Metadata Standard,” Preliminary Review Version 1.0, April 2003
RPG BUILD 2.5	VV&A Recommended Practices Guide, http://vva.dmso.mil/
06S-SIW-115	From FOMs to BOMs and Back Again
05S-SIW-084	Understanding the BOM Metadata and Making It Work For You
05S-SIW-088	RPR-BOM Initiative: Providing a Set of Applicable BOMs to the M&S Community
04F-SIW-024	The Application of Base Object Models (BOMs) for Supporting Aggregation and Load Balancing
04F-SIW-052	The Application of Base Object Models (BOMs) for Enabling Multi-Resolution Modeling
04S-SIW-111	Applying Design Patterns for Enabling Simulation Interoperability
03F-SIW-080	Capturing Intent-Of-Use Meta-Data for the Conceptual Model – A Key to Component Reuse
03F-SIW-109	Manufacturing BOMs with SRML for Process-Oriented Federations
03S-SIW-031	A Common Framework for Military M&S and C4I Systems
03S-SIW-143	Case Study: Prototyping a Mega-BOM with SRML for Next Generation Combat
02F-SIW-038	The Simulation Reference Markup Language (SRML): A Foundation for Representing BOMs and Supporting Reuse
02F-SIW-100	The Use of Base Object Models (BOM) in the Virtual Backlot (VB)
99S-SIW-115	Object Model Use Cases: A Mechanism for Capturing Requirements and Supporting BOM Reuse
99S-SIW-185	A Structural Description of Base Object Models (BOMs)
98F-SIW-034	Base Object Models (BOMs): Reusable Component Objects for Federation Development

* The document associated to this reference may be replaced with any superseding document that is considered an update.

3 Definitions

The definitions identified Table 3-1 are common terms used in this document. Given that the meaning of some of these terms differs among domains of interest, these definitions are provided to identify the meaning of the terms in the scope of this document. The Authoritative Dictionary of IEEE Standards Terms should be referred for terms not defined in this section.

Table 3-1 Common Terms

Term	Definition
Abstraction	An idea that concentrates on the essential properties of a common *pattern of interplay* or *state machine* rather than on specific realizations or actual cases.
Base Object Model	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.
BOM Assembly	A composition of BOMs that can result in a Federation Object Model (FOM), Simulation Object Model (SOM), or pattern which encompasses a larger scope.
BOM Component Implementation	A component containing the functionality described by a BOM for a specific language or platform.
Component	"Reusable building blocks which have a known set of inputs and provide expected output behavior, but the implementation details may be hidden. Such components are useful for constructing simulations and/or providing functionality for simulation systems." – Community of Interest (COI) M&S Metadata Focus Group
Composability	"The ability to rapidly select and assemble components to construct meaningful simulation systems to satisfy specific user requirements. Composability includes the framework, body of knowledge, tools, techniques, and standards necessary to enable effective integration, interoperability, and reuse." – DoD M&S Master Plan
Conceptual Entity	An abstract representation of a real world entity, phenomenon, process, or system. Here after referred to as *entity*.
Conceptual Event	A representation of a transient action that occurs among *conceptual entities* that may affect the *state* of one or more of the *conceptual entities*. Here after referred to as *event*.
Conceptual Model	A description of "what the [simulation or federation] will represent, the assumptions limiting those representations, and other capabilities needed to satisfy the user's requirements."[1]
Federate	A simulation, an interface to a live system, or a supporting utility such as a Logger, Plan View Display, or Stealth Viewer that can interoperate with other such software systems in a federation. In HLA, a federate is "an application that may be or is currently coupled with other software applications under a Federation Object Model Document Data (FDD) and runtime infrastructure (RTI)."[2]

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

[2] IEEE 1516-2000, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules, March 2000.

Federation	A collection of one or more federates capable of interoperating within a distributed synthetic environment. In HLA, “a federation is a named set of federate applications and a common Federation Object Model (FOM) that are used as a whole to achieve some specific objective.”[3]
Glyph	An image used for the visual representation of a BOM such as within a tool palette or web page.
Message	An *event* that is directed to a specific receiver(s).
Metadata	“Structured, encoded data that describe characteristics of information-bearing entities to aid in the identification, discovery, assessment, and management of the described entities."[4]
Pattern	A named set of recurring behavior used to accomplish a common objective, capability, or purpose.
Pattern Action	A single step in a *pattern of interplay* that may result in a *state* change of a *conceptual entity*. A *pattern action* can be represented by either a defined *event* within the BOM or by another BOM.
Pattern of Interplay	Specific type of pattern characterized by a sequence of *pattern actions* involving one or more *conceptual entities*.
Simulation Space	Any part of a simulation environment that may be represented by one or more *conceptual entities* modeled and reflected by a federate, federation, or as an aggregate entity within a federation execution.
State Machine	A description of the various *states* or conditions of a *conceptual entity*, and how the *pattern actions* associated with one or more *patterns of interplay* may affect these conditions over the *conceptual entity’s* life.
Trigger	An *event* that is not directed to a specific receiver(s).

[3] IEEE 1516-2000, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules, March 2000**.**

[4] The Final Report of the Association for Library Collections and Technical Services’ Task Force on Metadata (2000).

4 Acronyms and Abbreviations

The following acronyms and abbreviations are used in this document.

API	Application Program Interface
BOM	Base Object Model
BCI	BOM Component Implementation
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance
COI	Community of Interest
CPU	Central Processing Unit
DDMS	Defense Discovery Metadata Specification
DIF	Data Interchange Format
DLL	Dynamic Link Library
DoD	Department of Defense
DSO	Dynamic Shared Object
DTD	Document Type Definition
EXCOM	Executive Committee
FDD	Federation Object Model (FOM) Document Data
FEDEP	Federation Development and Execution Process
FEDEX	Federation Execution
FOM	Federation Object Model
HLA	High Level Architecture
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers, Inc.
M&S	Modeling and Simulation
OMT	Object Model Template
OS	Operating System
PDG	Product Development Group
POC	Point of Contact
RAD	Rapid Application Development
RFOM	Reference Federation Object Model
RPG	Recommended Practice Guide
RTI	Run-time Infrastructure
SAC	Standards Activity Committee
SISO	Simulation Interoperability Standards Organization
SOA	Service Oriented Architecture
SOM	Simulation Object Model
SRML	Simulation Reference Markup Language
TAD	Technical Area Director
UML	Unified Modeling Language
VCL	Visual Component Library
VV&A	Verification, Validation, and Accreditation
WSDL	Web Services Description Language
XML	Extensible Markup Language
XMSF	eXtensible Modeling and Simulation Framework
XSD	XML Schema Definition

This page left intentionally blank

5 BOM Rationale

M&S continues to provide a proven utility for testing, training, and scientific analysis. Its application is expanding in many arenas and disciplines including military and homeland defense, education, manufacturing, medical, logistics, aviation, environmental science, and more. SISO seeks to promote interoperability and reuse within the M&S community and explore ways to enable composability for these arenas and disciplines. BOMs serve to address the operational and technical needs in these areas, especially in regards to composability.

5.1 Issues

Typically, the development and deployment of simulations and simulation environments requires a significant amount of time, effort, and collaboration. Expending these resources might be permissible if it was only occasionally required to develop and deploy simulations and simulation environments, but this is not the case. M&S is pervasive and exists across many different domains and interests. For example, within the military domain, simulation is an important aspect of training, testing, analysis, mission rehearsal, and prototyping. However, simulation is also used widely outside of the military domain. Today, a distributed simulation exercise may often be at a Joint level where combined assets have been assembled with the intent to model and test them in a large simulation environment.

Some current technology enablers include the following:

- Interoperability standards such as the HLA and the Run-time Infrastructure (RTI) to connect systems and allow them to exchange meaningful data
- Process standards such as the Federation Development and Execution Process (FEDEP) which help provide guidance and ensure objectives are being met
- Various tools and Application Program Interfaces (APIs) that make the job easier for simulation software designers and system engineers to build and test federates within a distributed simulation environment

Despite these enablers, the task in building and putting together simulation and simulation environments, which must conform to common agreed upon message interfaces, remains an arduous task for the M&S community at large. It takes a long time, a great deal of effort, and much collaboration among participants, which include sponsors, simulation software designers, system engineers, and testers. What is required is a composable infrastructure that encourages the development and reuse of components that are matched to the needs of the desired simulation or simulation space, across the entire community. A BOM component standard is seen as an enabler for supporting composability.

5.2 Benefit of a BOM Framework

The BOM framework as documented in the BOM specification and this BOM guidance document is intended to influence the following eight capabilities within the M&S community:

1) **Interoperability** – The application of Extensible Markup Language (XML) and XML Schemas prescribed for BOMs provide a mechanism for defining and validating context, and facilitates understanding of the data being exchanged. Furthermore, the flexibility offered by BOMs allows greater application of simulation interoperability within other domains.

2) **Reusability** –The metadata cataloged within a BOM (such as intent-of-use, and integration use history) coupled with the conceptual model information a BOM may provide (such as *patterns of interplay* and *state machines*) will help to facilitate greater reuse of components.

3) **Composability** – BOMs make possible the ability to rapidly compose simulations and simulation environments both statically (at design time) and dynamically (at run-time).

4) **Adaptability** – BOM Assemblies produced by BOM compositions can be used to represent the standard data exchange interface for systems, simulations, and their components. For instance, HLA compliant federates can continue to use their specific BOM Assembly interface to experiment within environments comprised of other simulations and systems represented by their own unique BOM Assembly interface. Adaptability may be accomplished by deploying and applying, by a receiving federate, the appropriate XML-based transformations that represent mappings among common BOMs within a BOM Assembly.

5) **Aggregation** – The application of BOMs can be used for supporting two types of aggregation: Model Aggregation and Entity Instance Aggregation. Model Aggregations reflect the coupling of related BOMs that can be grouped to form the interface of a much larger model or system that is to be represented for a federation. For instance, a BOM assembly of individual BOMs 1) arrival and seating, 2) ordering meal, and 3) restaurant payment could be combined into an assembly for restaurant dining. This type of aggregation would typically occur prior to exercise execution. Entity Instance Aggregations reflect the coupling of multiple entities into a single inclusive group, and would likely be applied during a federation execution (FEDEX). Essentially, a BOM would be used to represent a container interface of a common group of entities. For instance, a BOM or BOM Assembly representing a battalion might be produced to represent an isolated group of troops and their associated vehicles and equipment. The benefit of either type of aggregation is that it can help reduce the amount of traffic distributed over the exercise network and reduce the load upon a system.

6) **Multi-resolution Models** – BOMs can be used to represent the behavior needed for modeling a *conceptual entity* associated to one or more *patterns of interplay*. Multiple BOM Component Implementations (BCIs) can be developed that represent the entities at differing levels of resolution. Federates can potentially choose from an assortment of BCIs that best represents their needs and system capabilities, and still adhere to common interface representations defined by BOMs or BOM Assemblies. Because of the use of this common interface reflected by the BOM, it is possible for BCIs of varying resolutions to be used and exchanged by a federate before the start of an exercise or dynamically during an exercise, assuming that proper precautions are taken to ensure validity and consistency. The benefit of multi-resolution models in this manner is that it allows for the optimization of federates and federations and can help reduce the Central Processing Unit (CPU) load upon a system.

7) **Rapid Application Development (RAD) Tools and Web Services -** It is envisioned that the next generation of tools and Web services (such as collaborative development environments and reuse libraries) will emerge to support the creation, deployment, and use of BOMs for component and simulation development, maintenance, and run-time support.

8) **Service Oriented Architectures (SOAs)** - The trend to move from system specific functionality to SOAs is likely to have an enormous influence on distributed simulation. Computer grids are composing services to deliver functionality needed by grid users. Given that BOMs separate the interface and pattern of a service from its implementation, BOMs can be used to migrate from existing system centric solutions to SOA capable M&S services.[5] BOMs provide additional insight into the behavior of components used to implement them, beyond the limited interface description provided by technologies such as Web Services Description Language (WSDL), thus enabling semantically meaningful composition of components described by BOMs.

[5] http://www.webopedia.com/TERM/S/Service_Oriented_Architecture.html

5.3 Market Opportunities

The flexibility offered by an open BOM specification coupled with the RAD type capability it offers, lends itself to a myriad of domains and markets, which demand for this type of composability. This includes education and distance learning, medical and biotechnology, manufacturing and processing, game development and entertainment, environmental and space sciences, logistics and humanitarian efforts, and much more.

Simulation provides a cost effective tool for all these industries, and BOMs provide an efficient mechanism for enabling such simulation. Some anticipated applications of BOMs that should be considered include the following:

- Virtual “hands-on” learning
- Enabling Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) and M&S interoperability
- Supporting web-based simulations (such as the eXtensible Modeling and Simulation Framework (XMSF) profiles)
- Maintaining HLA compatibility
- Using it for rapid prototyping
- Leveraging BOMs (and BOMs metadata) for enabling the semantic web

This page left intentionally blank

6 BOM Concept

Previously it was stated that a BOM contains the essential elements for specifying aspects of the conceptual model and documenting an interface of a simulation component that can be used and reused in the design, development, and extension of interoperable simulations. Such a component is intended to be used as a building block in the development and extension of a simulation and/or a federation of simulations.

6.1 Component-Based Approach

As illustrated in Figure 6-1, the idea of component-based development has been a driving force of many other industries for quite some time.

Home Construction	**Plumbing and Electrical Supplies, Lumber, Hardware**
Electronics Engineering	**Integrated Circuits, Capacitors, Resistors**
Software Development	**VBX, ActiveX, VCL, and JavaBean Controls**

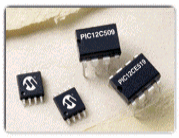

Figure 6-1 Examples of Component Reuse

To better understand the concept of BOMs, consider the home construction analogy from Figure 6-1. When a person sets out to build a new home, which is analogous to building a federation, they must first have a conceptual understanding of the major design features in the home and the relationships among those features. In the building industry, that conceptual understanding of features and relationships is commonly captured in architectural drawings, which is analogous to a conceptual model, and is then developed into detailed blueprints. These blueprints provide a suitable specification for the home's construction.

Although not a perfect analogy, the final blueprints for the home are comparable to the Federation Object Model (FOM) for a federation. A FOM specifies the major features of the federation (*HLA object classes* and *HLA interaction classes* with their associated *attributes* and *parameters*) and relationships among those features (the *HLA object class* and *HLA interaction class* hierarchies). Within a blueprint what is revealed are many common elements or "patterns" of a home design. For instance, the elements required of a bathroom, which typically feature a shower, bath, toilet, and sink, are common in most homes and are reflected in the blueprint in a common yet customized way. The same could be said for a garage, kitchen, family room, or bed room. These common elements or "patterns" can help expedite the development of a blueprint. This approach is analogous to the BOM concept. The BOM provides a mechanism to identify features and relationships of federates and federations with respect to the conceptual model. Within a BOM such features are identified as *conceptual entities*, and a feature's behavior is described within a *state machine*. Most importantly relationships among *conceptual entities* to be modeled by a federate or federation are described by *patterns of interplay*, which can be applied for various designs. The common behavior associated to a *pattern action* is identified by a *conceptual event,* or a more detailed BOM. These conceptual model features and relationships, which are captured in a BOM, are customized and specified within a FOM or BOM using HLA OMT class structures.

When selecting home features, the purchaser can choose one of the following approaches (with the comparable analogies for FOM construction):

(1) The home purchaser can select to use an existing blueprint. In terms of HLA federation construction this is analogous to selecting an existing FOM. A reference FOM is a common choice in these situations.
(2) The home purchaser can work with the architect and engineer to develop a completely new set of architectural drawings and blueprints. In federation construction, this is analogous to building the FOM from scratch, sometimes referred to as a top-down approach to FOM construction.
(3) The home purchaser can work with the architect and engineer to combine elements from existing, proven designs. For example, the purchaser may select the layout for a set of bedrooms, bathrooms, closets, and hallways from one set of designs; the design of the living and dining rooms from another set; and the design of kitchen, laundry room, and garage from a third set. Each one of these sets may exist prepackaged in a design system. This third option most closely resembles the construction of a FOM from component pieces (i.e., construction from BOMs representing *patterns of interplay*). In the home construction and the FOM construction, some customization will be needed to integrate the design components into the final product specification.

Following the selection of an approach (see items 1, 2 or 3 above), the building components identified in the design, such as fixtures, appliances, flooring, hinged doors, and windows can be selected. At the conceptual level, these building components identify the functionality necessary to equip a building or home and are roughly analogous to BOMs. Like building components called out in an architectural drawing, BOMs describe the essential capabilities and features needed at the conceptual level for a federate or federation. Furthermore, just as building components can be used for a variety of commercial or residential projects, BOMs are intended to provide useful design information for a variety of federate and federation needs.

Although other approaches exist for the design of a home, the third approach (see item 3 above) provides a sufficient contrast illustrating the use of blueprint subsets as a component mechanism for building a total home blueprint. Likewise BOMs offer a component mechanism that can be used for building a FOM.

The full range of flexibility within the M&S community is needed to properly meet the demands of its users. Having ready-made, individual BOMs and BOM Assemblies for simulation software designers and system engineers to compose into various custom combinations allows interoperability technology such as HLA to be effectively applied throughout the entire M&S community. The impact of this approach can be best understood by comparing the impact component-based development has had on other markets such as software development, electronics engineering, and home construction. In short, these markets have been revolutionized by the component-based approach. The BOM methodology offers a similar impact for the M&S interoperability community.

6.2 Composability

Composability relates to the assembly of components that meet specific needs which can be combined to establish complete, validated simulations and simulation environments. This idea of composability is comparable to how integrated circuits are used to build electronic products, or how building components are used to build or extend homes and buildings as illustrated previously in Figure 6-1.

The enabling technology anticipated to provide a component-based mechanism for composability within the M&S community is the BOM. Like software components such as ActiveX controls, Visual Component Library (VCL) controls, or Java Beans, which are all used to support the rapid application development (RAD) of software applications, BOMs are intended to provide a distinct way to represent individual simulation interplay patterns and components that can be used and combined to build, modify, or extend the representation and capability of a federate or federation. This concept of applying BOMs that support a specific need in creating BOM compositions is illustrated in Figure 1-1.

The intent of BOMs is to reduce the time and effort associated with the development, modification, and generation of federates, federations, and even aggregations. Thus, rather than constructing federation interfaces from scratch or from the extraction/merging of pre-existing Simulation Object Models (SOMs), and/or FOMs, it is possible for such interfaces to be constructed more rapidly and efficiently from prebuilt, reusable BOMs. This principle of coupling BOMs into larger grained interfaces can also be applied to the development and enhancement of federates.

6.3 BOM Architecture Overview

The diagram illustrated in Figure 6-2 depicts the basic architecture views of the simulation space for federates and federations. The purpose of the layered view is to convey the effects that the BOM standard has on the M&S HLA community. This figure also shows the concept of libraries of object models and BOMs to facilitate reuse.

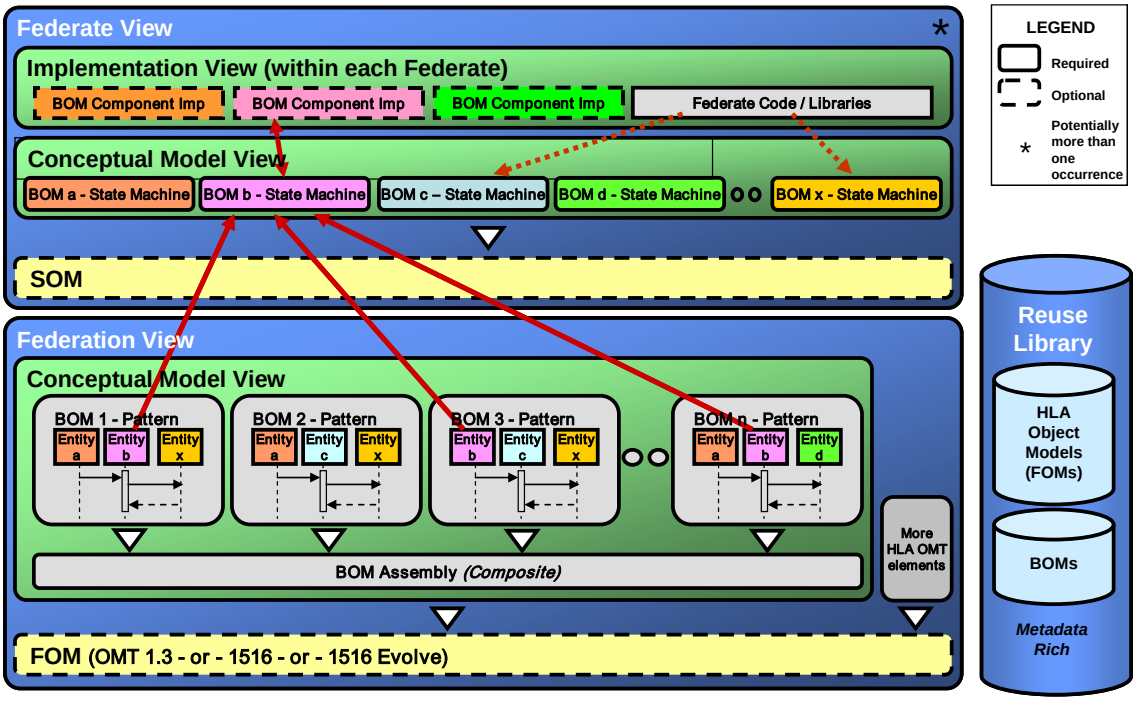

Figure 6-2 Federate / Federation Architecture View

In the HLA, the Federation View is typically supported by an HLA FOM for identifying the type of data that will be conveyed among the players of the federation during an execution. It is at the Federation View that BOMs and BOM Assemblies (composites of BOMs) can be used to help define the features of the FOM in a component-based manner. At this level, BOMs are used to identify specific *patterns of interplay* that are to be represented within a federation. The *conceptual entities* and *events* supporting the *pattern of interplay* can be associated (mapped) to one or more *HLA object classes*, *HLA interaction classes*, and their *attributes* and *parameters*. These mapped Object Model Definition elements may be found either within the current BOM or found externally within another BOM, FOM or SOM.

A BOM Assembly is formed by a collection of BOMs resulting in the creation of a higher order pattern that identifies the set of *entities*, *events*, *pattern actions*, and supporting classes to be represented by the federation. In principle, a BOM Assembly is a composite interface much like an HLA FOM is an interface; they both are

capable of exposing the *HLA object classes*, *HLA interaction classes*, *HLA attributes*, and *HLA parameters* to be shared among the federation. Because of the common heritage with the HLA OMT Specification, a BOM Assembly can be used to generate a FOM. A BOM Assembly, however, provides a unique mechanism for encouraging modularity over a FOM and provides a richer set of metadata.

At the top of Figure 6-2 is the Federate View, which reveals the capability each federate has for supporting the requirements of the federation. Within the Federate View, there is a Conceptual Model View and an Implementation View. At the Conceptual Model View, BOMs can be used to identify the necessary behavior needed for modeling a *conceptual entity* within a federate using a *state machine*. Each *conceptual entity* can be associated (mapped) to one or more *HLA object classes*, *HLA interaction class*, and their *attributes* and *parameters*, which are found either within the current BOM or found externally within another BOM, FOM or SOM. The Implementation View represents the code or component implementations needed by the federate to model the *conceptual entities* and *state machines*, and support the *conceptual events* and *patterns of interplay* expected by the federation.

At the right of the figure 6-2 is a Reuse Library, which reveals the capability to identify and select appropriate models, including BOMs, for fulfilling the goals and objectives of a federation.

The Federation View and the Federate View are the two architectural views related to BOMs that can be applied to support composability, and the Reuse Library can be applied to support the reuse of BOMs in supporting composability for either federates or federations. The Federation View is characterized by the composition of BOMs, which describe the *patterns of interplay* and *conceptual events* to be modeled by the federation. The composition of BOMs is reflected through a BOM Assembly, which can be used to generate an HLA FOM. The Federate View is characterized by BOMs describing the *state machines* and *conceptual entities* to be modeled by each federate. The supporting *HLA object classes* and *HLA interaction classes* that map with a BOM's *conceptual entities* and *conceptual events* are required for both federates and federations. In the Federation View, these class structures can be used to generate an HLA FOM. In the Federate view, these class structures can be used as the foundation for code and component implementations.

6.4 Application of BOMs for Providing Conceptual Level Support

As identified earlier, BOMs serve to provide an end-state of a simulation conceptual model and can be used as a foundation for the design of executable software code and integration of interoperable simulations. The BOM provides a Conceptual Model view allowing key aspects of a simulation conceptual model to be captured in a BOM. This includes static descriptions of items resident in the real world, which are captured in terms of *entity types* and *event types*, and how these *entity types* and *event types* relate or interact with each other in the real world, which are captured in terms of *patterns of interplay* and *state machines*. Both these static and interaction views of a conceptual model described using a BOM are useful when the simulation software designers begin to consider what their simulation will need to do.

6.5 Application of BOMs for Providing Federation Level Support

As illustrated in Figure 6-3, a BOM Assembly is used to identify a composite interface, which can be used to represent a federate, federation, or even an aggregation of patterns and federate object instances within the simulation space. Other types of interfaces commonly known in the community include a SOM or a FOM.

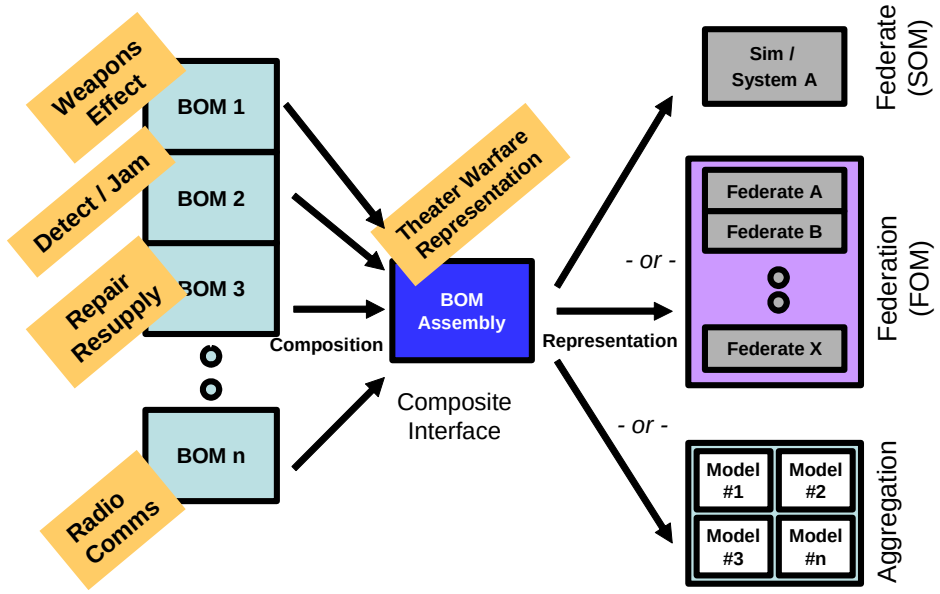

Figure 6-3 Application of BOMs for Providing Various Types of Federation Level Support

Within a BOM Assembly, *metadata* is associated with each integrated BOM element including *purpose* and integration *use history*. The dependency and interrelationships between BOMs is captured at the Conceptual Model Definition level of a BOM Assembly within a *pattern of interplay.* Within this *pattern of interplay,* each independent *BOM* used to collectively define the BOM Assembly may be identified by a *pattern action,* or *variation*. Furthermore, a BOM Assembly representing a composition of BOMs provides a clear delineation of interface elements to be represented at the federation level in the context of the Object Model Definition view, which includes *HLA object classes* and *HLA interactions classes.* Such interface elements can be extracted, altered, swapped or extended by other BOMs.

6.6 Application of BOMs for Providing Federate Level Support

It is the responsibility of the federate to model the behavior and characteristics codified by a BOM that it uses. One approach to ensure the simulation can model the behavior and characteristics identified by a BOM is to use a matching component model that implements the required behavior of the BOM at the desired resolution. Figure 6-4 illustrates how the execution of a BOM Component Implementation (BCI) at run-time can be used to fulfill the capabilities of the BOM within the federate. Examples of BCIs include a Java Bean, a Windows Dynamic Link Library (DLL), a Unix Dynamic Shared Object (DSO), an Active X component, a Simulation Reference Markup Language (SRML) module which contains JavaScript, or a Flash component which contains ActionScript.

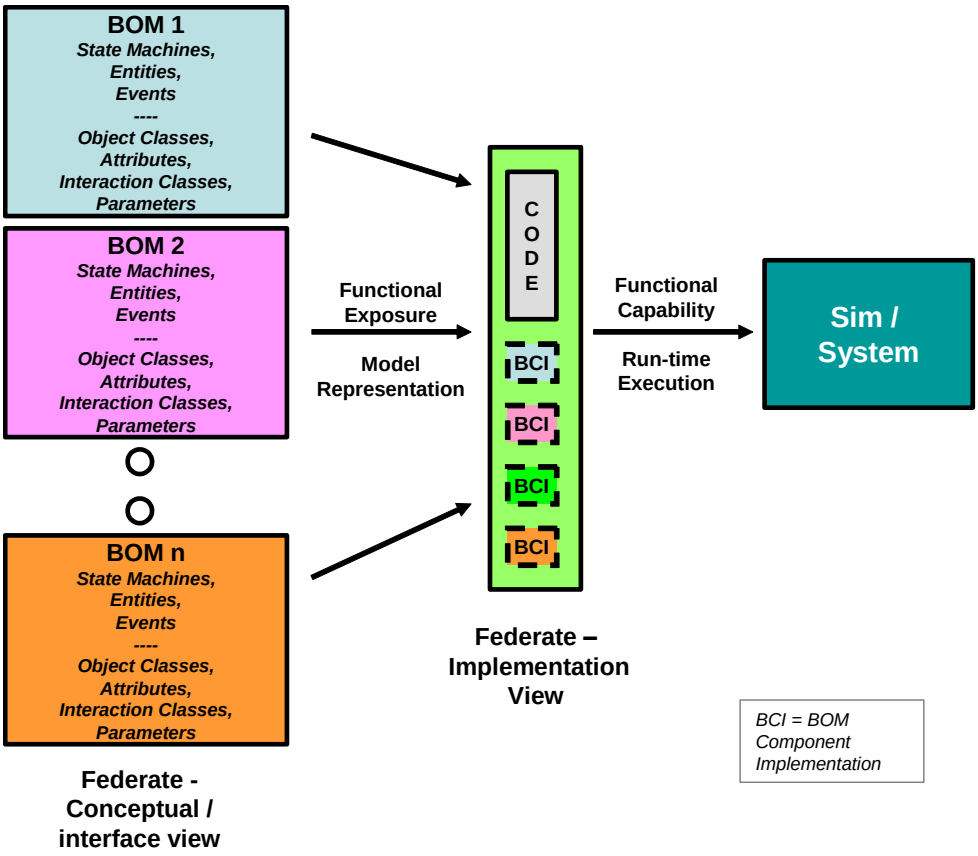

Figure 6-4 Application of BOMs for Providing Federate Level Support

It should be noted that federates and any simulation environments derived from BOMs and represented by BOM Assemblies do not require the use of BCIs. If the functional capability to be modeled at run-time is already an intrinsic element of the federate's behavior, (i.e., it is coded within the federate), then the use of a BCI is not necessary. However, if a federate lacks a specific behavior and the functional capability can be found within a BCI that matches the characteristics of a BOM, then this would be an opportunity to use the matching BCI rather than implementing new code within the federate.

7 The Role of the BOM within the FEDEP

The HLA Federation Development and Execution Process (FEDEP) defines a generalized process for constructing distributed simulation environments (federations) from reusable simulation components (federates). The FEDEP has seven basic steps, as illustrated in Figure 7-1.

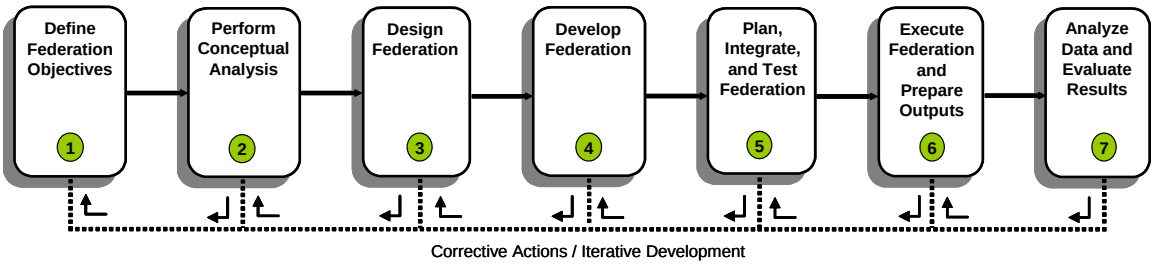

Figure 7-1 FEDEP Seven-Step Process

To meet the diverse needs of a very wide community of users, the FEDEP is designed as a framework rather than as a prescriptive process. That is, rather than defining a "one size fits all" methodology for all users, the FEDEP focuses on defining the fundamental activities and high-level information flows necessary to build an HLA federation, and leaves the actual implementation approach to the discretion of the development team. This provides the flexibility necessary for new and creative approaches to federation development problems/issues to emerge and to be shared across user domains.

The application of BOMs within the federation development process defines one such approach. BOMs represent an innovative mechanism for facilitating composability that fits extremely well within the FEDEP framework. As each FEDEP step is described, relevant aspects of the BOM concept can be applied as appropriate to achieve desired outcomes. For example, FEDEP step 2 defines the need to "develop an appropriate representation of the real world domain that applies to the federation problem space." Although the FEDEP provides a high level description of a conceptual model and why it is important in the overall process, it does not define how it should be implemented. Applying a BOM-based implementation approach, a basic analysis of functional requirements and the domain(s) of interest is performed to discover relevant *patterns of interplay*. The conceptual model is then structured as a pattern-based characterization of the problem space. This type of structuring fosters the identification of applicable BOM components later in the process.

Other examples of how BOMs can be successfully applied within the FEDEP framework are easily identified. For instance, in FEDEP steps 3 and 4, BOMs can be mapped to required *patterns of interplay*, and then assembled into a BOM Assembly as required to meet overarching federation objectives. The BOM Assembly then serves as the foundation for the application FOM. Also in step 4, BOMs can provide appropriate behavior support for federate implementations, which are then integrated into federations in FEDEP step 5.

Throughout all steps, the intent is to increase the speed and efficiency by which the federation is constructed by leveraging reusable "building block" components rather than having to start from a clean slate for each new application.

In summary, although the FEDEP defines the overarching framework for the construction of HLA federations, it does not explicitly define the lower-level methodologies and supporting technologies needed to perform the various activities identified within the process model. The BOM concept leverages the FEDEP framework to define a wholly unique approach to rapid federation construction, with the depth of detail necessary to support federation implementations across many different user domains. Sections 8 and 9 will expand upon the fundamental concepts introduced in this section.

8 BOM Development

BOMs provide the framework for representing a piece part of a conceptual model, simulation object model or federation object model. This includes *patterns of interplay*, *state machines*, *entity types*, *event types*, *HLA object classes*, and *HLA interaction classes* for both simulations and simulation environments. This section explores how to begin the creation of an individual BOM. For those individuals who wish to simply use BOMs in building and/or representing simulations and simulation environments, skip to Section 9.

There are two basic approaches for developing BOMs that provide federation level support. Either a BOM can be built from scratch or it can be built by extracting piece part elements from an existing federation, individual federate, FOM, or SOM. As illustrated in Table 8-1, a BOM intended to represent a conceptual model to be reflected among federates within a federation must contain *metadata* within the Model Identification view, and, must contain a *pattern of interplay*, *state machine*, *entity type*, and/or *event type* within the Conceptual Model Definition view. Whereas, a BOM intended to represent a class structure used by a federate, must contain *metadata* within the Model Identification view, and, *HLA object classes*, *HLA interaction classes*, *HLA attributes,* or *HLA parameters* within the Object Model Definition view, The BOM Assembly applies the same template (as described in Section 9), with the exception of the need for the Object Model Definition and Model Mapping. Object Model Definition and Model Mapping are optional but highly recommended for all BOM types.

Table 8-1 BOM Elements for Representing an Individual BOM

BOM Element ->	Model Identification	Conceptual Model	Model Mapping	Object Model Definition
Individual BOM (Conceptual Model) behavior description elements (pattern, state machine, events)	Required	Required	Optional	Optional
Individual BOM (Object Model Definition) interface description elements using HLA OMT constructs (object classes, interaction classes)	Required	Optional	Optional	Required

The Model Identification provides the essential *metadata* needed so that the BOM can be discovered and properly reused. The Conceptual Model Definition identifies the capabilities to be represented in terms of *patterns of interplay, state machines, conceptual entities,* and *conceptual events*. The Model Mapping provides an association between *conceptual entities* and *events* in the Conceptual Model Definition with the *object classes* and *interaction classes* found in the Object Model Definition. Finally, the Object Model Definition table provides the means for identifying the various *object classes*, *interaction classes*, *attributes,* and *parameters* that can support the elements needed to represent the BOM.

8.1 Model Identification

The purpose of the Model Identification view is to document certain key identifying information within the BOM description itself. It provides a minimum but sufficient degree of descriptive information about a BOM. For instance, when simulation software designers and system engineers wish to pose detailed questions to those responsible for the development and distribution of the BOM, point-of-contact (POC) information within a BOM is important. This helps facilitate understanding, which leads to reuse.

For any resource to be reusable, it must be visible and accessible. By visibility, it is not only meant that the ability exists to recognize that BOMs are available, but also that they have sufficient descriptors to allow a potential user to determine whether an existing BOM is appropriate for his/her use. To that end, a common set of descriptors are defined in the Model Identification view of the BOM. The second factor in reusability, accessibility, will come when libraries of BOMs are constructed, and their descriptions are captured in a consistent format (commonly called *metadata*).

The *metadata* that can be contained in the BOM Model Identification view is drawn from a commonly used set of metadata standards including the IEEE Std 1516.2-2000 HLA OMT Specification, which hereinafter will be referred to as the HLA OMT Specification. The Object Model Identification Table found within the HLA OMT Specification has been adopted for the BOM Model Identification view, and has been extended to include elements from other common standards such as the Dublin Core, Defense Discovery Metadata Specification (DDMS), and the Verification, Validation, and Accreditation (VV&A) Recommended Practice Guide (RPG).

The BOM Model Identification Table is illustrated in Figure 8-1.

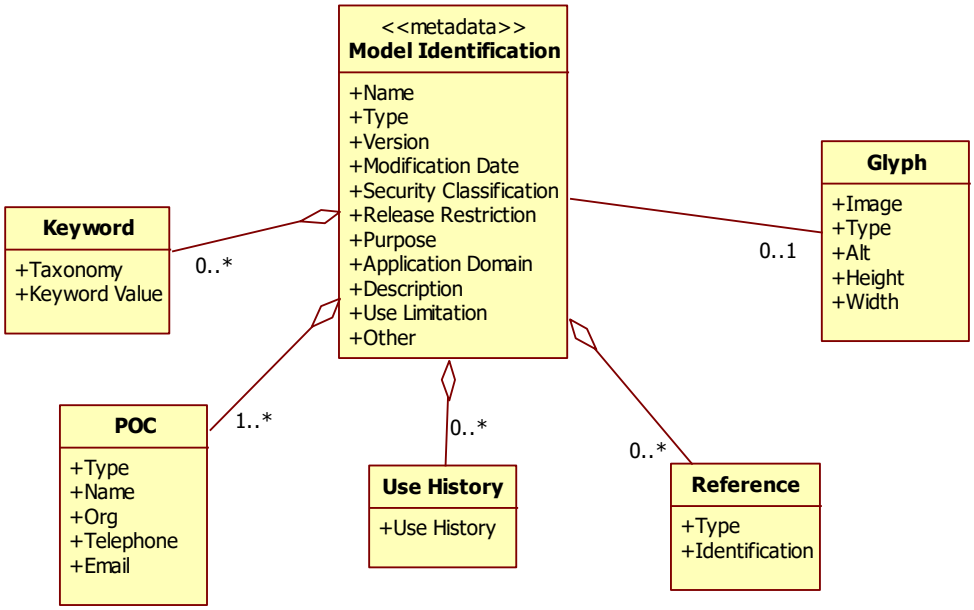

Figure 8-1 BOM – Model Identification Table Element Relationship

Table 8-2 provides an example use of the BOM Model Identification Table for documenting *metadata*.

Table 8-2 – Weapon Effects Model Identification Metadata

Category	Information
Name	Weapons Effect
Type	BOM
Version	1.0 Beta
Modification Date	2005-06-04
Security Classification	Unclassified
Release Restriction	Not for release outside the XYZ Larson Missile Defense
Release Restriction	Release only to SISO community members
Purpose	RPR FOM decomposition

Category	Information
Application Domain	Real-time Platform Simulation
Description	This is an example to illustrate the key concepts of a BOM
Use Limitation	None
Use History	Used as example in BOM Guidance document
Keyword	
Taxonomy	Military Warfare
Keyword Value	Engagement
POC	
POC Type	Primary author
POC Name	D. Vader
POC Organization	Empire
POC Telephone	+1 44 123-456-7890
POC Email	dvader@empire.com
POC	
POC Type	Release authority
POC Name	
POC Organization	XYZ Larson Missile Defense
POC Telephone	+1 44 123-456-1000
POC Email	the.lawyer@xyz-defense.com
Reference	
Type	Glossary
Identification	ISBN 12345678901
Other	None
Glyph	
Type	JPG
Alt	WeaponsEffect
Height	32
Width	32
Note	na

The key in defining BOM *metadata* is to ensure that enough descriptive content has been captured regarding a BOM's application, so that it can be properly reused. The following bullets identify the guidance necessary for consistent application of the BOM *metadata*. Where no guidance is provided, the explanations located in the BOM Specification are assumed to be self-explanatory:

- Name – The *name* of the BOM should be descriptive and unique. It is unlikely that any central authority will be available to assign unique identifiers for every BOM that is created. Thus, it is important that the BOM name be sufficiently descriptive to allow a user to locate the BOM by keyword searches while differentiating it from other BOMs of a similar nature.

- Type – Use this field to indicate what specific *type* of model is being defined. For BOMs, the value of this field is "BOM."

- Version – Use the local versioning identification system used by the BOM author. The important concern is that a potential user should be able to differentiate among multiple versions of the same BOM. If the BOM undergoes a revision changing the *version*, then the *modification date* should be updated as well.

- Purpose – Use this text field to identify the sponsor's / developer's intended *purpose* for the BOM. BOM users may determine that a BOM is useful for purposes other than the one for which it was initially intended. However, this metadata will still be useful when searching out BOMs for reuse.

- Use Limitation – The author of the BOM will use this field to identify any known constraints associated to a BOMs potential use within a federate or simulation environment. The intent is to minimize the

number of inappropriate applications of a BOM for unintended purposes and to reduce the number of incorrect assumptions that may have been made if the selection of a BOM was based solely on other *metadata* values including its *name*, *purpose* or *keywords*.

- Use History – The author of the BOM will initially populate this field. However, to maximize reuse of a BOM, other users who have found the BOM useful should provide feedback to the author of the BOM so that he/she can list the additional *use history*. Evidence of broad applicability and frequent reuse will generally be gauged by potential users as evidence of BOM quality. The *use history* field should identify the version of the BOM to which the *use history* comment applies. (See Section 9.6 for more information).

- Keywords – BOM authors are urged to associate as many *keywords* with a BOM as is practical. The *keywords* may come from a list of the author's own creation. More useful will be lists of *keywords* from lists established by communities of interest.

- Point of Contacts (POCs) – Each BOM should have at least one *poc*. Usually, this will be the author of the BOM. However, the template allows for multiple *pocs* to be associated with a BOM. In line with the *use history* guidance above, *pocs* could be established in user organizations.

- References – The use of multiple *references* is encouraged. One common *reference* would be the conceptual model documentation from which a BOM was developed (see Section 8.2). Other *references* might identify related BOMs or identify related FOMs and/or SOMs where the BOM was employed. The *reference type* specifies the way in which the *reference* is related to the BOM.

- Other – This field can contain any information that the author of the BOM thinks is relevant. Although not prohibited, the author should avoid using this field for information that fits one of the other metadata fields.

- Glyph – This is an *image* that can be used to represent a BOM on a tool palette or web-based reuse library. It is suggested that unique images are used to represent a BOM to differentiate between other BOMs. The graphic should be as simple as possible while still conveying the intended use of the BOM and differentiate the BOM from others.

8.2 The Conceptual Model

Before creating a BOM, conceptual model analysis identifying the operational objectives, functional requirements, and the various patterns and states to be represented should be performed.

The activities associated with capturing and cataloging the conceptual model within the BOM match closely with FEDEP steps 1 and 2, as illustrated in Figure 8-2. This includes developing and using federation objectives statement (1.2) and producing a federation conceptual model (2.2), which can be used to aid in the selection and reuse of BOMs and carried forward within the BOM through several mechanisms provided by the BOM template. The resulting conceptual model should provide a description of what the simulation component, simulation or federation "will represent, the assumptions limiting those representations, and other capabilities needed to satisfy the user's requirements."[6]

Aspects of the conceptual model can be documented within the BOM through the use of the *purpose* and *reference* fields provided within the Model Identification view, and through the use of the *pattern of interplay*, *state machine*, *entity type,* and *event type* template components provided in the Conceptual Model Definition view.

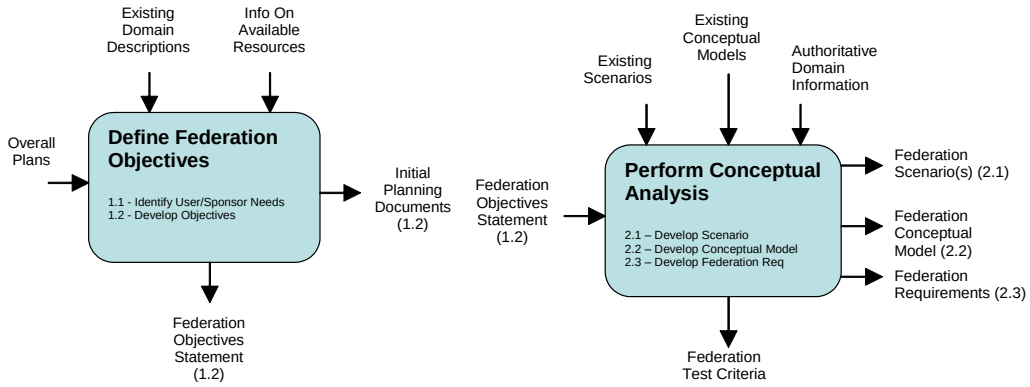

Figure 8-2 FEDEP Steps 1 & 2

The discernable attributes of a conceptual model are defined as follows:

- Describes functional and behavioral capabilities
- Maps to objectives / stakeholder requirements
- Identifies conceptual entities to be represented
- Identifies logic and algorithms
- Identifies relationships
- Identifies assumptions
- Identifies limitations[7]

The Model Identification Table of a BOM provides a mechanism for identifying *purpose,* which may be reflective of objectives and stakeholder requirements, and identifying *references* to information such as a comprehensive

[6] IEEE 1516.3, IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP), 2002**.**

[7] 03F-SIW-080, "Capturing Intent-of-Use for the Conceptual Model - A Key to Component Reuse", Gustavson, Zimmerman, Turrell

conceptual model. Methods that can be used to physically *reference* the conceptual model to the BOM include the following:

1) The use of an internal reference, where a representation of the conceptual model is embedded or appended within the BOM Document.
2) The use of an external reference (for example a hyperlink), where a representation of the conceptual model is contained outside of the BOM Document.

Additionally, the Conceptual Model Definition provides a mechanism to identify up to the following four different template components for representing the needs of a simulation.

- *Pattern of interplay*
- *State machine*
- *Entity type*
- *Event type*

8.2.1 Pattern of Interplay

Aspects of the conceptual model can be reflected in the BOM's *pattern of interplay* template component. The relationship of elements within this component is illustrated in Figure 8-3.

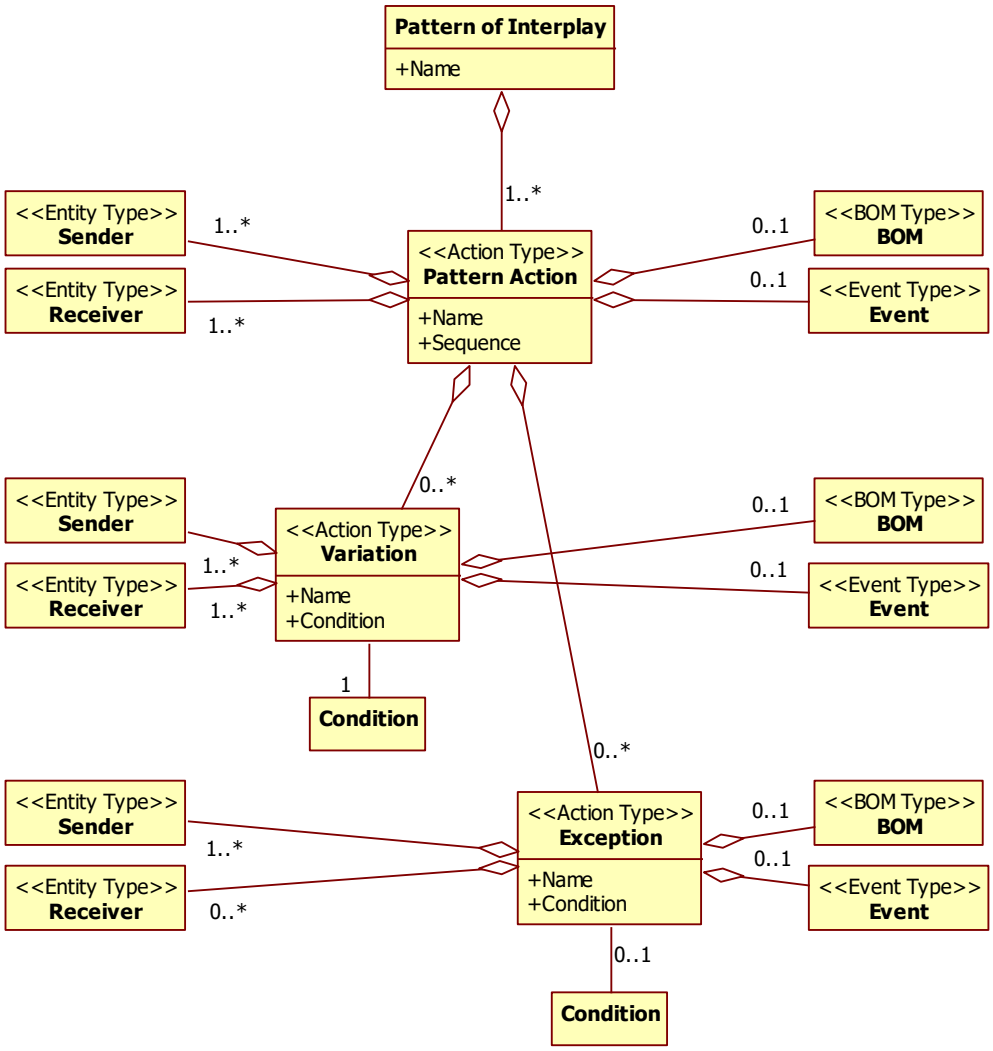

Figure 8-3 Pattern of Interplay Element Relationship

A *pattern of interplay* is represented by one or more *pattern actions* needed to accomplish a specific purpose or capability. Each *name* associated with a *pattern action* provides a title reflective of the functionality required to carry out that *pattern action*. For each *pattern action*, one or more *senders* and *receivers* are identified to provide a means for understanding the behavioral relationship among *conceptual entities*, which are defined by *entity types* (see Section 8.2.3). The activity required for fulfilling each *pattern action* can be associated with either an *event type* (see Section 8.2.4), or another *BOM*. This latter aspect provides the basis for creating a BOM Assembly from a composition of BOMs and is described in Section 9.

The *pattern of interplay* also supports the ability to identify *variations* and *exceptions* that may occur for any one *pattern action*. A *variation* is used to identify a different way that an action can be achieved. An *exception* is used to identify any unexpected but potential behavior for an action.

An example of a BOM *pattern of interplay* is provided in Table 8-3.

Table 8-3 Weapon Effects Pattern of Interplay

Pattern of Interplay Name		Seq	Name	Sender	Receiver	Event	BOM	Condition	Note
Restaurant Payment	**Pattern Action**	1	Weapon Fire	FiringEntity	TargetEntity	WeaponFire	na		na
	Pattern Action	2	Munition Detonation	FiringEntity	TargetEntity	Munition Detonation	na		na
	Pattern Action	3	Damage State Update	TargetEntity	FiringEntity	DamageStateUpdate	na		na
Note	na								

Weapons Effects is a *pattern of interplay* that is prevalent in many simulations and federations. The *conceptual entities* include a shooter, identified as **FiringEntity**, and a target, identified as **TargetEntity**. The activities between these *conceptual entities* include a **WeaponFire** and, if successful, a **MunitionDetonation**, which results in a **DamageStateUpdate** by the target. The interface details of how each *pattern action* is carried out are dependent upon an *event* and/or associated *BOM* linked with the *pattern action* provided in the table. This example contains no *variations* or *exceptions*.

Unified Modeling Language (UML) can be used to help convey and understand the information contained within a BOM. Figure 8-4 illustrates this Weapons Effects *pattern of interplay* using a UML sequence diagram. Whereas UML provides a visualization technique, the BOM Template Specification provides a familiar mechanism for capturing the behavior information of *patterns of interplay*, *states*, *entities,* and *events* in a context that is appropriate for supporting but not limited to HLA.

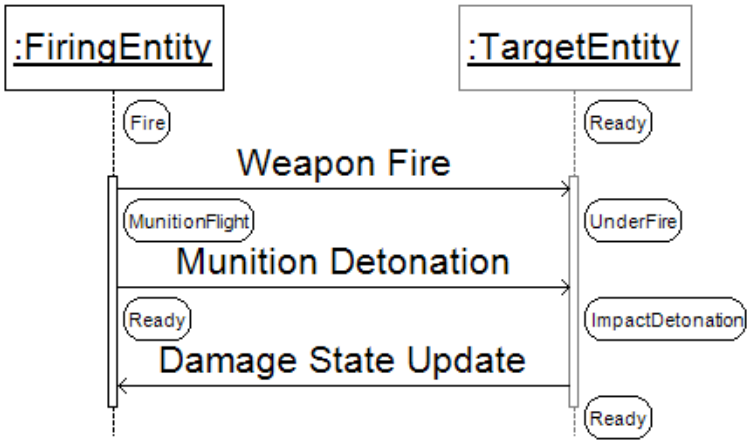

Figure 8-4 Weapons Effects Pattern of Interplay

The vertical lines in the above diagram represent the lifetime of the *conceptual entity* labeled at the top. The horizontal lines represent the *pattern actions* that occur among these *conceptual entities*. Additionally, this particular sequence diagram has been decorated with the *states* for both the *sender* identified as **FiringEntity** and *target* identified as **TargetEntity** as rounded boxes beside each *entity's* lifetime.

8.2.2 State Machine

The *state machine* template component is used to identify the needed behavior *states* anticipated of a *conceptual entity* to support one or more *patterns of interplay*. The *state machine* may use a *pattern action* from a *pattern of interplay* as an *exit action* to transition from the current *state* to the *next state*. Figure 8-5 provides an illustration of the *state machine* organization.

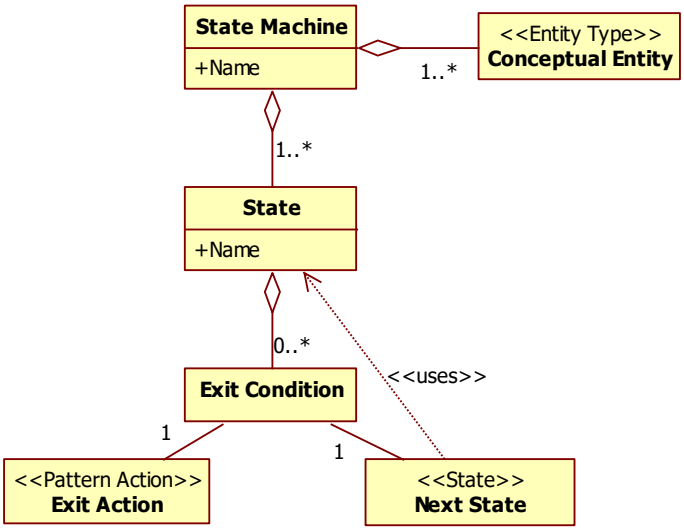

Figure 8-5 State Machine Element Relationship

A *state machine* can be associated with one or more *conceptual entities* that fulfill the role defined by the states. The *state machine* supports one more states where each *state* may transition to a subsequent *state,* which is identified as *next state*, upon an *exit action*, which is associated with a *pattern action* identified in a *pattern of interplay*.

Using the BOM Template Specification, the *state machine* for describing the roles of the "shooter" within the Weapons Effects BOM example is shown in Table 8-4.

Table 8-4 State Machine for Firing Entity

State Machine Name	Conceptual Entity	State			Note
		State Name	Exit Condition		
			Exit Action	Next State	
ShooterStates	FiringEntity	Ready	CommandToFire	Fire	na
		Greet	WeaponFireAction	MunitionFlight	na
		Seat	MunitionDetonationAction	Ready	na
Note	na				

In this example, the only *conceptual entity* identified for fulfilling the *state machine* is the **FiringEntity**. The details of a *conceptual entity*, such as **FiringEntity**, are defined separately within a BOM as described in Section 8.2.3. Each state transition is carried out by an *exit action*. Details of the *exit actions*, such as **WeaponFireAction**, and **MunitionDetonationAction**, including their dependency on an *event type* or *bom* to carry out the *pattern action*, maybe provided elsewhere in the BOM using the *pattern of interplay* template component (see Section 8.2.1).

Figure 8-6 illustrates the **state machine** describing the role for the "shooter" of the Weapons Effects BOM using a UML state diagram.

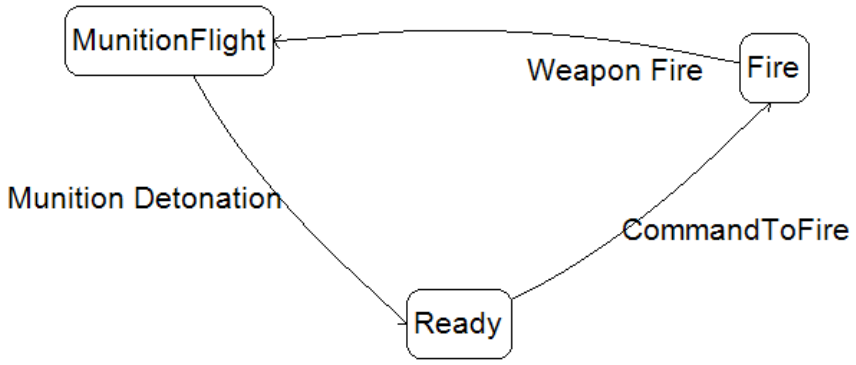

Figure 8-6 State Machine for the Firing Entity of a Weapon Effects BOM

Each block identifies a *state*, and the message lines between states represent the *pattern action* that takes place, thereby satisfying the *exit condition*.

As another example, the *state machine* for describing the roles of the "target" within the Weapons Effects BOM example is shown in Table 8-5.

Table 8-5 State Machine for Target Entity

State Machine Name	Conceptual Entity	State			Note
		State Name	Exit Condition		
			Exit Action	Next State	
TargetStates	TargetEntity	Ready	WeaponFireAction	UnderFire	na
		UnderFire	MunitionDetonationAction	ImpactDetonation	na
		ImpactDetonation	DamageStateUpdateAction	Ready	na
Note	na				

In this example, the only *conceptual entity* identified for fulfilling the *state machine* is the **TargetEntity**. Again, the details of a *conceptual entity*, such as **TargetEntity**, are defined separately within a BOM as described in Section 8.2.3. Each state transition is carried out by an *exit action*. Details of the *exit actions*, such as **WeaponFireAction**, **MunitionDetonationAction**, and **DamageStateUpdateAction**, including their dependency on an *event type* or BOM to carry out the *pattern action*, maybe provided elsewhere in the BOM using the *pattern of interplay* template component (see Section 8.2.1).

Figure 8-7 illustrates the *state machine* describing the role for the "target" of the Weapons Effects BOM using a UML state diagram.

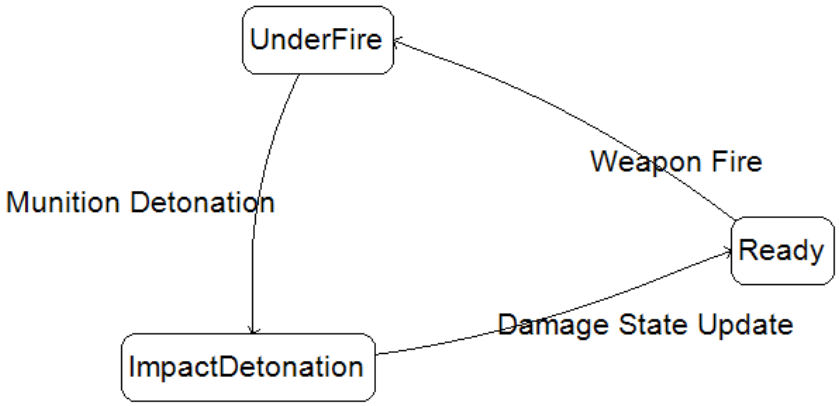

Figure 8-7 State Machine for the Target Entity of a Weapon Effects BOM

Again, each block identifies a state, and the message lines between states represent the *pattern action* that takes place, thereby satisfying the exit condition.

8.2.3 Entity Types

A key to understanding both *pattern of interplay* and *state machine* elements described previously is to focus on the *conceptual entities* being characterized. A *conceptual entity* is an abstract representation of a real world entity, phenomenon, process, or system. These *conceptual entities* are needed to understand the relationships within a *pattern of interplay*, the roles with respect to *state machines* across one or more *patterns of interplay*, and the responsibilities as *sender* and/or *receiver* in regards to the *events* that can occur to fulfill a *pattern of interplay*.

Aspects of the conceptual *entity types* are reflected in the BOM's *entity type* structure as illustrated in Figure 8-8. These aspects include the *entity characteristics* associated with each *entity type*.

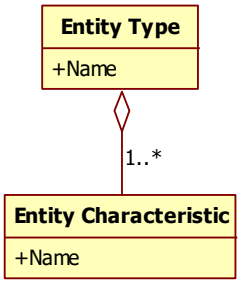

Figure 8-8 Entity Type Element Relationship

As an example, Table 8-6, which uses the BOM Template Specification, identifies two *conceptual entities*: one called **FiringEntity** and the other called **TargetEntity**. The *entity characteristics* for these two *entity types* are also identified in the table.

Table 8-6 Entity Table Example

Entity Type Name	Entity Characteristic Name	Note
FiringEntity	ID	na
	Location	na
TargetEntity	ID	na
	Location	na
	Velocity	na
Note	na	

Lexicon definitions can be included for *entity types* as illustrated in Tables 8-7 and 8-8.

Table 8-7 BOM Entity Type Definition Table Example

Type	Definition
FiringEntity	Entity that fires a weapon at a target
TargetEntity	Entity that is the intended target of a weapon fire

Table 8-8 BOM Entity Type Characteristic Definition Table Example

Type	Characteristic	Definition
FiringEntity	ID	Unique id for firing entity
	Location	Physical position of the firing entity
TargetEntity	ID	Unique id for target entity
	Location	Physical position of the target entity
	Velocity	Velocity for the entity

8.2.4 Event Types

Since the BOM can be used to identify a complete *pattern of interplay*, additional information is needed to document how the *pattern of interplay* takes place. Consider that each *pattern of interplay* consists of a sequence of *pattern actions* that take place to achieve a specific capability, objective or purpose. Each *pattern action* is supported by either an *event* or another *bom*. An *event* can be either a *trigger* or *message*. Aspects of the *conceptual events* are reflected in the BOM's *event type* structure as illustrated in Figure 8-9. These aspects include *characteristics* associated with each *event type* and the role the *characteristics* play in supporting the *event*. These roles include *source*, *target*, and *content.* Also identified is the *trigger condition*, which is expressed in the form of a Boolean expression.

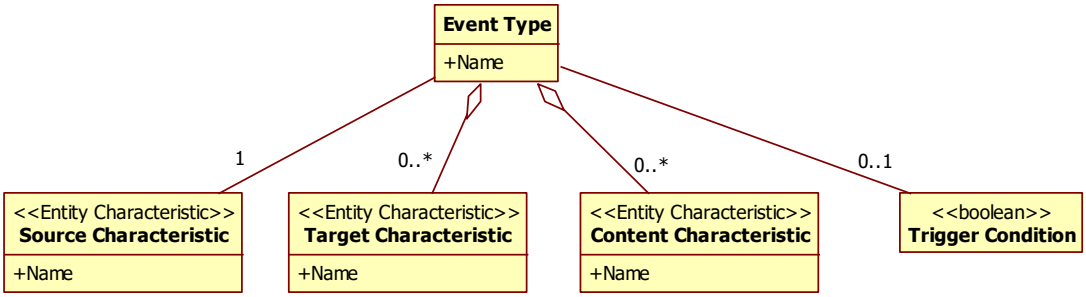

Figure 8-9 – Event Type Element Relationship

The types of BOM *events* used to represent and carry out a *pattern action* are identified as *triggers* and *messages*. The difference between a *trigger* and a *message* is that a *trigger* is an undirected exchange of information and *messages* are directed exchanges of information (e.g. someone telling something to someone else) where the *sender* knows about the intended *receiver* of the *message*. Both of these concepts are illustrated respectively in Figure 8-10 and Figure 8-11.

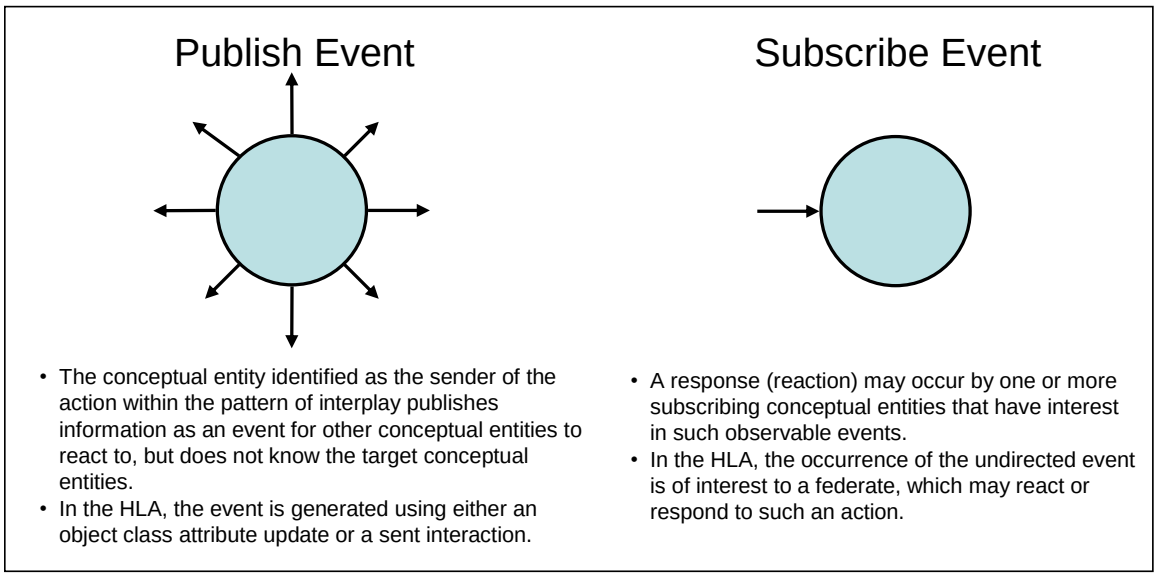

Figure 8-10 Event – Trigger

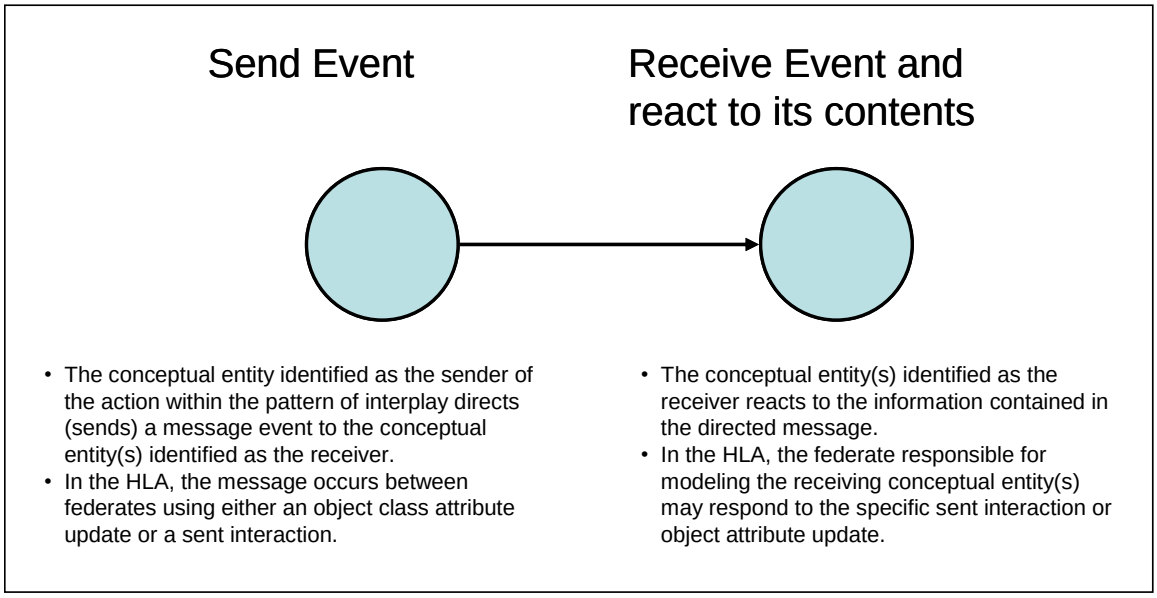

Figure 8-11 Event – Message

As an example, Table 8-9, which uses the “Base Object Model (BOM) Template Specification,” defines three *conceptual events*, which support the *pattern of interplay* identified in Table 8-3 and described by the *metadata* in Table 8-2.

Table 8-9 Weapons Effect Event Type Table

Entity Type Name	Source Characteristic Name	Target Characteristic Name	Content Characteristic Name	Trigger Condition	Note
WeaponFire	FiringEntity_Identifer	TargetEntity_Identifier	Munition_identifer	na	na
MunitionDetonation	FiringEntity_Identifer	TargetEntity_Identifier	na	na	na
DamageStateUpdate	TargetEntity_Identifier	na	na	Damage_sustained == TRUE	na
Note	na				

In this example, the first two *events*, **WeaponFire** and **MunitionDetonation**, are both representative of *messages* intended for a specific *conceptual entity*. However, the third *event*, **DamageStateUpdate**, is representative of a *trigger* not intended for any specific *conceptual entity*, but rather intended to affect any *conceptual entities* that have interest in the *trigger condition*, which is that **Damage_sustained == TRUE** by the **TargetEntity**.

Lexicon definitions can be included for event types as illustrated in Table 8-10.

Table 8-10 BOM Event Type Definition Table Example

Type	Definition
WeaponFire	Message for representing weapon fire interaction among conceptual entities
MunitionDetonation	Message for representing weapon detonation interaction among conceptual entities
DamageStateUpdate	Trigger for representing entity update interaction among conceptual entities

8.3 Model Mapping

Model Mapping provides a mechanism for mapping between the *entity types* and *event types* of the Conceptual Model Definition (see Section 8.2) and the class structure elements of the Object Model Definition, which are described using HLA OMT constructs (see Section 8.4). There are two types of Model Mappings supported: *entity type mapping* and *event type mapping*.

8.3.1 Entity Type Mapping

The *entity type mapping* template component provides a mechanism for mapping between the *entity type* elements of the Conceptual Model Definition and the class structure elements of the Object Model Definition described as illustrated in Figure 8-12.

As an example, Table 8-11, which uses the BOM Template Specification, identifies the mapping for two *entity types* that were defined as part of the Conceptual Model Definition with supporting HLA *object classes* and HLA *interaction classes*. The mapping of the *characteristics* of the *entity type* with supporting HLA *attributes* and *parameters* is identified as well.

The *condition* value can be used to describe under what conditions an HLA *attribute* or *parameter* can be applied and/or used to represent a *characteristic* of a *conceptual entity* or *conceptual event*. However, if no *condition* value is known for the use and application of an HLA *attribute* or *parameter*, "na" shall be identified.

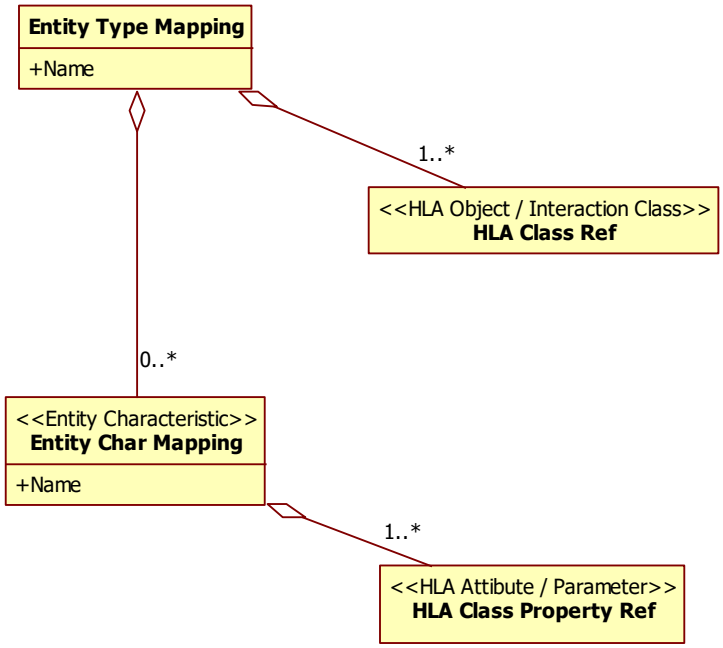

Figure 8-12 Entity Type Mapping Element Relationship

Table 8-11 Weapons Effect Entity Type Mapping Table

Entity Type Mapping		Entity Characteristic Mapping		Note
Entity Type Name	**HLA Object / Interaction Classes**	**Entity Char Name**	**HLA Attribs / Params**	
FiringEntity	HLAobjectClassRoot.BaseEntity.PhysicalEntity.Lifeform.Human	ID	Human.EntityIdentifer	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
Note	na			

8.3.2 Event Type Mapping

The *event type mapping* template component provides a mechanism for mapping between the *event type* elements of the Conceptual Model Definition and the *class structure* elements of the Object Model Definition as illustrated in Figure 8-13.

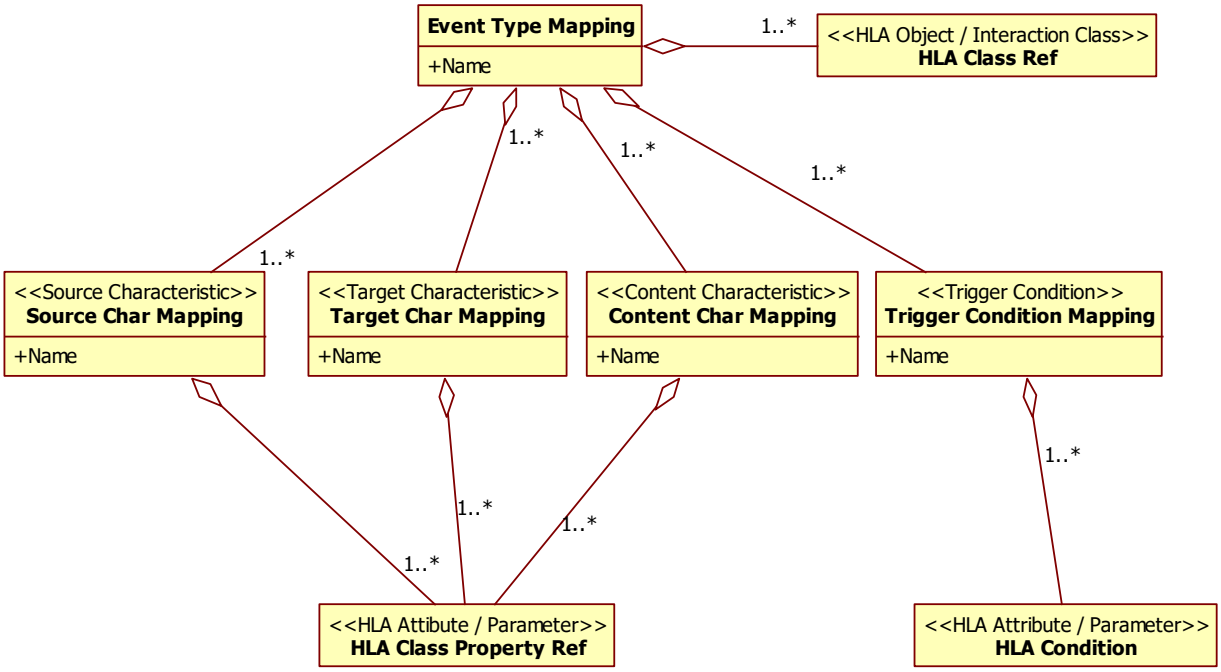

Figure 8-13 Event Type Mapping Element Relationship

As an example, Table 8-12, which uses the BOM Template Specification, identifies the mapping for three *event types* that were defined as part of the Conceptual Model Definition with supporting HLA *object classes* and HLA *interaction classes*. The mapping of the characteristics of the *entity type* with supporting HLA *attributes* and *parameters* is identified as well.

The *condition* value can be used to describe under what conditions an HLA *attribute* or *parameter* can be applied and/or used to represent a *characteristic* of a *conceptual event*. However, if no *condition* value is known for the use and application of an HLA *attribute* or *parameter*, "na" shall be identified.

Table 8-12 Weapons Effect Event Type Mapping Table

Event Type		Source Characteristic		Target Characteristic		Content Characteristic		Trigger Condition		Note
Name	**HLA Object / Interaction Classes**	**Name**	**HLA Attribs / Params**	**Name**	**HLA Attribs / Params**	**Name**	**HLA Attribs/ Params**	**Trigger Expres-sion**	**HLA Condition**	
WeaponFire	HLAinteractionRoot. WeaponFire	FirintEntity_ Identifier	HLAinteractionRoot. WeaponFire. FiringObjectIdentifier	TargetEntity_ Identifier	HLAinteractionRoot. WeaponFire. TargetObjectIdentifier	MunitionEntity_ Identifier	HLAinteractinRoot. WeaponFire. MunitionObject Identifier	na	na	na
MunitionDetonation	HLAinteractionRoot. MunitionDetonation	FirintEntity_ Identifier	HLAinteractionRoot. Detonation. FiringObjectIdentifier	TargetEntity_ Identifier	HLAinteractionRoot. Detonation. TargetObjectIdentifier	na	na	CustomerEntity. PaidBill = FALSE	Customer. hasPaid == FALSE	na
DamageStateUpdate	HLAobjectClassRoot. BaseEntity. PhysicalEntity	Damage sustained from weapon fire	HLAobjectClassRoot. BaseEntityPhysicalEntity. DamageState	WaiterEntity. ID	RequestBill. WaiterID	na	na	na	HLAobjectClassRoot. BaseEntity. PhysicalEntity. DamageState != previous HLAobjectClassRoot. BaseEntity. PhysicalEntity. DamageState	na
Note	na									

8.4 Object Model Definition

The Object Model Definition of a BOM is defined using HLA OMT constructs, which includes *HLA object classes*, HLA *interaction classes*, *attributes*, and *parameters* as illustrated in Figure 8-14. Additionally, *data types* can be defined to support these HLA class structures.

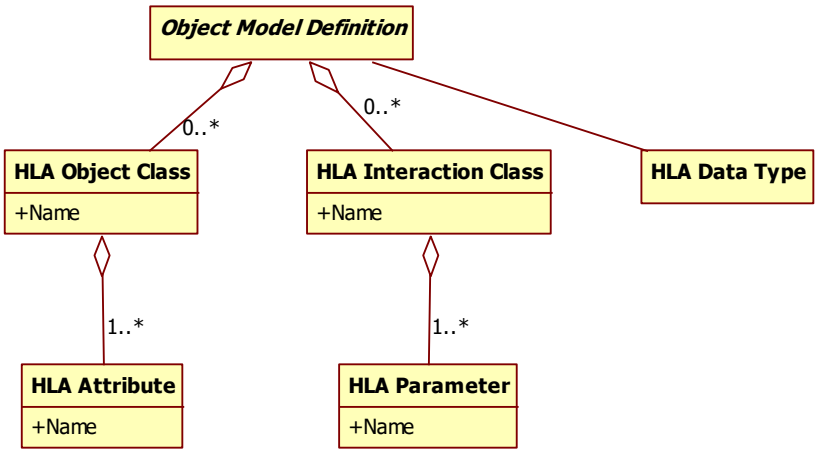

Figure 8-14 Object Model Definition Element Relationship

8.4.1 HLA OMT Class Structures

In HLA, the Object Model Template (OMT) is used to document certain federate or federation-relevant information, such as class-subclass relationships. Parts of the HLA OMT are prescribed by the "Base Object Model (BOM) Template Specification" to document how the BOM *entity types* and *event types* are implemented in HLA.

The HLA OMT Specification formally defines the valid structure of any HLA OMT file. To enable flexible reuse of the HLA OMT in the BOM Template, this structure has been captured as an XML Schema. Relevant pieces of the XML Schema are reused from the HLA OMT and are included as part of the BOM Template XML Schema.

Since the purpose of a BOM is to find suitable ways of implementing or supporting a *pattern of interplay* using HLA constructs, the BOM *events* such as *triggers* and *messages* are mapped to corresponding *HLA interaction classes* or *HLA object class attributes* so that an implementer can understand how BOM *triggers* and *messages* are carried out among federates.

In a BOM, the HLA OMT information is captured in the model definition. This includes *HLA object classes* and their *attributes*, *HLA interaction classes* and their *parameters*, *HLA data types*, and all other necessary information about how the data is encoded on the network when transmitted from one federate to another. Some information templates from the HLA OMT Specification are not used in the BOM Object Model Definition view, such as RTI switches, synchronization points, and user-defined tags.

The available values and default selections for filling the *HLA attribute* and *HLA parameter* fields are identified in Tables 8-13 and 8-14.

Table 8-13 HLA Attribute Values

	Attribute	Datatype	Update Type	Update Condition	Ownership	P/S	Available Dimensions	Transportation	Order
Available Values	*User defined*	*User defined*	Static, Periodic, Conditional, NA	*rate value*, on change, *condition value*, NA	D, A, N, DA, NA	P, S, PS, N, NA	NA	HLAreliable, HLAbesteffort, NA	Receive, Timestamp, NA
Default Selection	-	-	NA	NA	NA	NA	NA	NA	NA

Table 8-14 HLA Parameter Values

	Interaction	Parameter	Datatype	Available Dimensions	Transportation	Order
Available Values	*User defined*	*User defined*	Static, Periodic, Conditional, NA	NA	HLAreliable, HLAbesteffort, NA	Receive, Timestamp, NA
Default Selection	-	-	NA	NA	NA	NA

Tables 8-15 through 8-18 illustrate the use of the HLA OMT Specification templates for defining a BOM at the interface description level. These are referenced by the mapping tables illustrated in Section 8.3. Note, not all default selections described in Tables 8-13 and 8-14 used for these examples.

Table 8-15 Weapons Effect OMT Object Classes Table

HLA Object Class				
HLAobjectRoot	BaseEntity	PhysicalEntity	Platform	
			Lifeform	Human

Table 8-16 Weapons Effect OMT Object Class Attributes Table

Object	Attribute	Datatype	Update Type	Update Condition	Ownership	P/S	Available Dimensions	Transportation	Order
HLAobjectRoot	HLApriveldgeToDelete	NA	NA	NA	NA	NA	NA	HLAreliable	Receive
BaseEntity	Spatial	SpatialStruct	Conditional	NA	No Transfer	PS	NA	HLAreliable	Receive
PhysicalEntity	ForceIdentifiier	ForceIdentifierEnum8	Conditional	OnChange	No Transfer	PS	NA	HLAreliable	Receive
Platform	HatchState	HatchStateEnum32	Conditional	OnChange	No Transfer	PS	NA	HLAreliable	Receive
Lifeform	StanceCode	NA	Conditional	OnChange	No Transfer	PS	NA	HLAreliable	Receive

Table 8-17 Weapons Effect OMT Interaction Classes Table

HLA Interaction Classes	
HLAinteractionRoot	WeaponFire
	MunitionDetonation

Table 8-18 Weapons Effect OMT Interaction Class Parameters Table

Interaction	Parameter	Datatype	Available Dimensions	Transportation	Order
MunitionDetonation	MunitionObjectIdentifier	na	na	na	na
	RateOfFire	SpatialStruct	na	na	na
	MunitionType	ForceIdentifierEnum8	na	na	na
WeaponFire	TargetObjectIdentifier	HatchStateEnum32	na	na	na
	WarheadType	StanceCodeEnum32	na	na	na

8.4.2 HLA OMT Data Types

The attribute table and parameter table in the HLA OMT Specification both provide columns for data type specifications. The HLA provides the following data types which can be used: basic data type, simple data type table, enumerated data type table, fixed record data type table, array data type table, or variant record data type table. Many of these tables are illustrated in Tables 8-19 through Table 8-23.

Table 8-19 Basic Data Types

Name	Size	Interpretation	Endian	Encoding
HLAoctet	8	8-bit value	Big	Assumed to be portable among hardware devices
HLAinteger16BE	16	Integer in the range [–2^15, 2^15 – 1]	Big	16-bit two's complement signed integer. The most significant bit contains the sign
HLAfloat32BE	32	Single-precision floating-point number	Big	32-bit IEEE normalized single-precision format (see IEEE Std. 754-1985)
HLAfloat64BE	64	Double-precision floating-point number	Big	64-bit IEEE normalized single-precision format (see IEEE Std. 754-1985)
HLAinteger32BE	32	Integer in the range [–2^31, 2^31 – 1]	Big	32-bit two's complement signed integer. The most significant bit contains the sign

Table 8-20 Simple Data Types

Name	Representation	Units	Resolution	Accuracy
HLAASCIIchar	HLAoctet	NA	NA	NA

Table 8-21 Enumerated Data Types

Name	Representation	Enumerator - Name	Enumerator - Value
ParameterTypeEnum32	HLAinteger32BE	ArticulatedPart	0
		AttachedPart	1
ArticulatedPartsTypeEnum32	HLAinteger32BE	Other	0
		Rudder	1024
		…	…
StationEnum32	HLAinteger32BE	Nothing_Empty	0
		Fuselage_Station1	512
		…	…
DetonationResultCodeEnum8	HLAoctet	Other	0
		EntityImpact	1
		…	…
FuseTypeEnum16	HLAinteger16BE	Other	0
		IntelligentInfluence	10
		…	…
WarheadTypeEnum16	HLAinteger16BE	Other	0
		CargoVariableSubmunitions	10
		…	…
ArticulatedTypeMetricEnum32	HLAinteger32BE	Position	1
		PositionRate	2
		…	…

Table 8-22 Fixed Record Data Types

Name	Datatype	Cardinality	Encoding
HLAASCIIstring	HLAASCIIchar	Dynamic	HLAvariableArray

Table 8-23 Array Data Types

Name	Encoding	Field	Datatype
ArticulatedParameterStruct	HLAfixedRecord	ArticulatedParameterChange	HLAoctet
		Padding	HLAoctet
		PartAttachedTo	HLAinteger16BE
		ParameterValue	ParameterValueStruct
ParameterValueStruct	HLAfixedRecord	ArticulatedParamterType	ParamterTypeEnum

		ArticulatedParts	ArticulatedPartsStruct
		AttachedParts	AttachedPartsStruct
AttachedPartsStruct	HLAfixedRecord	Station	StationEnum32
		StoreType	EntityTypeStruct
EntityTypeStruct	HLAfixedRecord	EntityKind	HLAoctet
		Domain	HLAoctet
		CountryCode	HLAinteger16BE
		Category	HLAoctet
		Subcategory	HLAoctet
		Specific	HLAoctet
		Extra	HLAoctet

Members of these data type tables may contain members of other data type tables in order to create arbitrarily complex data types. As prescribed in the HLA OMT Specification, the name of any data type or data representation shall be unique among all data types and data representations. More information regarding the HLA OMT data types, which are used to support the BOM template components, can be found in the HLA OMT Specification.

8.5 Common Design Considerations

When developing BOMs, there are many design considerations. An important issue is in deciding upon the level of generalization that should go into the BOM design. The question to be asked by the simulation software designer and/or system engineer is, “Will the BOM be specific to one particular type of simulation interplay thread (i.e., concrete), or will it be robust by scaling it back to support multiple forms (representations) of a common simulation interplay thread (i.e., abstract)?” The answer to this question directly impacts the breadth of reuse. As the level of abstraction increases, so does the likelihood that the BOM can be applied for wider use. Essentially, BOMs can range anywhere from being a concrete BOM, identifying specific class types supporting a unique capability, to an abstract BOM, flexible enough to support a number of common pattern manifestations.

A concrete BOM can almost be used “as is,” with little customization needed -- just plug it in. The disadvantage, however, is that concrete BOMs are restricted to specific, detailed interplay requirements; therefore, they are less likely to be reused across differing domains.

However, by scaling the details back and making a BOM more general and abstract, the BOM can be used more frequently by a greater number of implementers. The drawback is that the implementation of an abstract BOM will require a larger degree of customization to support a specific federate user’s needs. When the BOM is implemented in a simulation or federation, the implementer must specify concrete classes. The concrete classes must be based upon the BOM they are derived from, reflecting its *attributes* or *parameters*. Thus, abstraction is desirable, but abstract BOMs will require greater customization to suit specific needs.

It is possible to also define concrete BOMs that result from abstract BOMs, thus creating layered BOMs. Therefore, if a developer is tempted to write a concrete BOM, he might consider starting with an abstract BOM and then supplement it with a concrete customization.

9 BOM Integration

Simulation software designers and system engineers benefit from the use of BOMs because of their ability to support rapid composition of simulations and simulation environments. BOMs should be chosen that support an aspect of the desired conceptual model for the intended simulation or simulation space. The collection of BOMs can then be used in the construction of an elevated BOM called a BOM Assembly. A BOM Assembly is a composition of BOMs integrated to form an aggregate or composite pattern. This support for a hierarchy of BOMs through BOM Assemblies can be used to represent various capabilities of a simulation or a simulation space, and, in support of HLA, can be used to generate a FOM or SOM.

Figure 9-1 provides a view of the activities associated with identifying the conceptual model, finding BOMs that match the conceptual model, developing BOMs representing a conceptual model and/or object model, and integrating BOMs to create BOM Assemblies. These activities map to FEDEP steps 2 through 4.

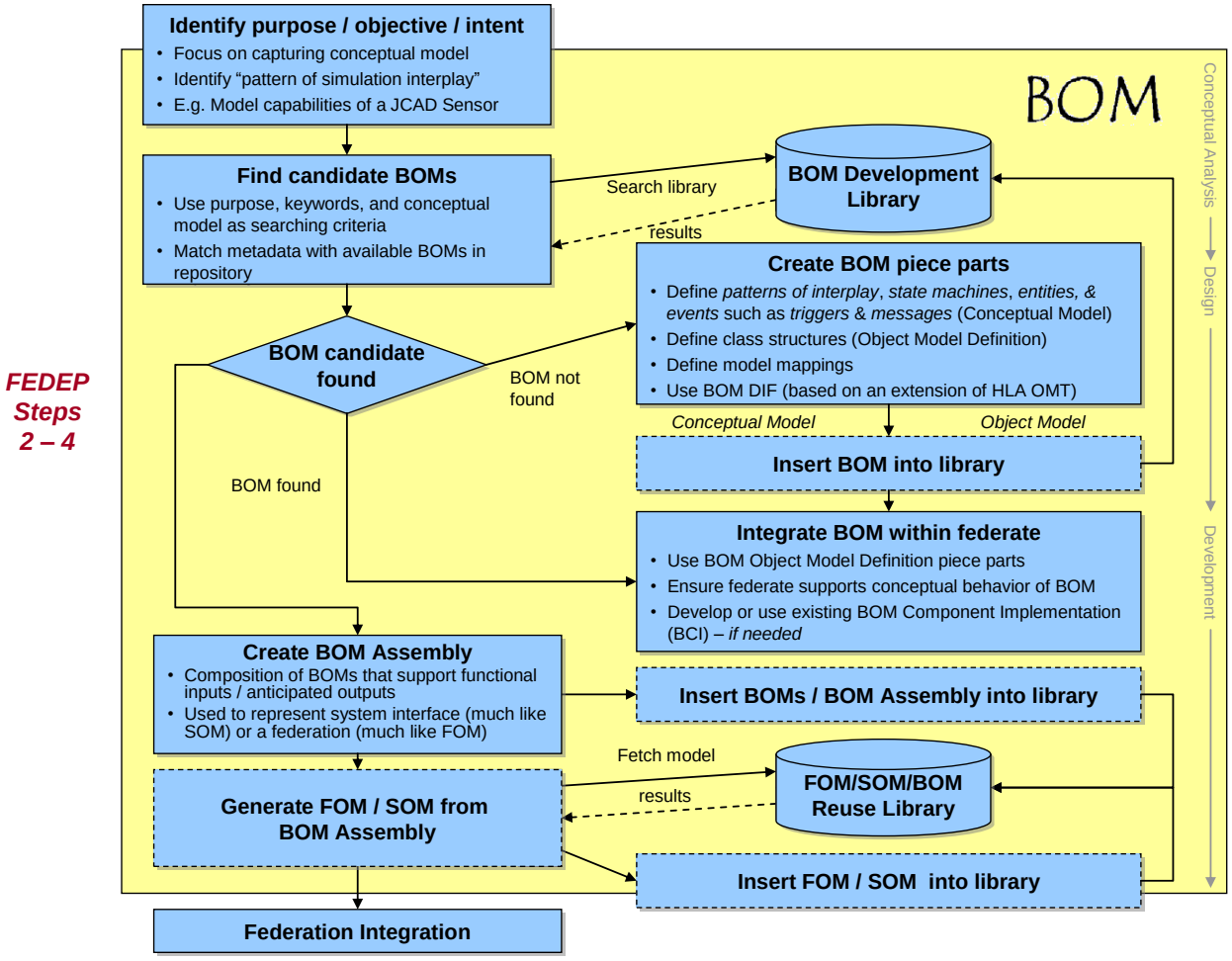

Figure 9-1 BOM Activities Related to the FEDEP

While a BOM Assembly can be used to represent a composite interface of a federation much like a FOM, the difference is that a BOM Assembly carries with it additional metadata that is absent within a FOM or SOM. The resulting BOM Assembly and its metadata can also be used to support later FEDEP phases. The metadata includes elements such as intent ("purpose") and integration experience ("use history"). Furthermore, the BOMs that make up a BOM Assembly allow greater support for configuration management

and modularity since the components and piece-parts that provide functionality for a federate or federation can be easily cataloged by the interface provided by the BOM.

As illustrated in Table 9-1, a BOM Assembly must contain a Model Identification view and a *pattern of interplay* within the Conceptual Model view. A BOM Assembly should have associated with it the *metadata* from each integrated BOM. Two ways of creating a BOM Assembly include copying the contents of multiple BOMs into a single BOM, or defining a higher order *pattern of interplay* in which the *pattern actions*, and or *variations* of a *pattern action* reference other BOMs. In the case of copying BOMs, either *metadata* is included in the copy or the original BOM metadata is provided as a *reference* within the BOM Assembly. Section 9.4 provides further discussion on integrating BOMs into a BOM Assembly.

Table 9-1 Elements for Representing a BOM Assembly

BOM Element ->	Model Identification	Conceptual Model	Model Mapping	Object Model Definition
BOM Assembly (representing composite of other BOMs)	Required	Required (Pattern of Interplay)	Optional	Optional

NOTE: The remainder of this section describes many of the issues that need to be addressed or resolved before attempting to select and use a BOM. This section is oriented toward the potential end-user of BOMs. For information on creating a BOM, see Section 8.

9.1 Reuse Approaches

There are two key approaches for using BOMs: Clean Sheet Development and Second Addition Development.

The Clean Sheet Development approach is building a BOM Assembly from scratch using BOMs as add-in model patterns or components. The Second Addition Development approach is simply using BOMs to supplement an existing BOM Assembly. Table 9-2 illustrates the summary efforts required for each of these two reuse approaches.

Table 9-2 BOM Reuse Approaches

CHOICES	Clean Sheet Development	Second Addition Development
GUIDELINES	• Capture requirements • Identify unique threads of patterns of interplay • Identify matching BOM based on identified patterns of interplay • Build new classes that support pattern needs • Capture states within BOM for simulation/system	• Analyze requirements • Recognize individual federate needs required • Identify matching BOM based on identified patterns of interplay • Build new classes that support pattern needs • Capture states within BOM for simulation/system

It is important to note that these two approaches are often contingent upon access to proper tools, BOM reuse libraries, and understanding the importance of the metadata needed to support the various BOM dimensions.

9.2 The Conceptual Model

The most important element for either BOM reuse approach is to “know what you’re looking for.” This is where one should never underestimate the importance of requirements gathering. In keeping with the FEDEP Model, which provides a generalized process for building and executing federations, it is best to begin by defining your federation objectives and to include what must be accomplished to achieve those objectives. The activities associated with step 1 of the FEDEP are depicted in Figure 9-2.

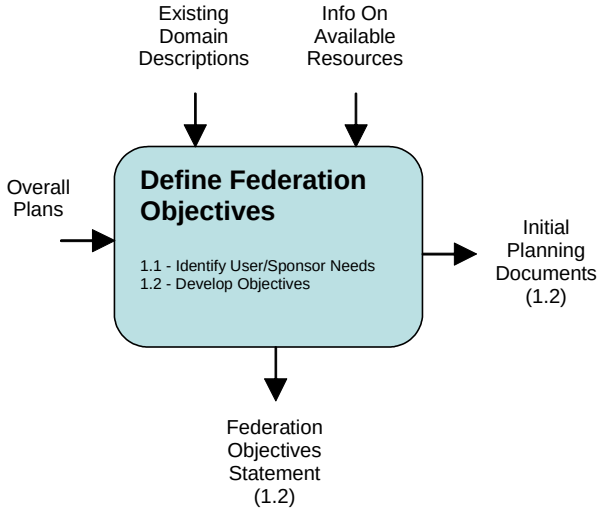

Figure 9-2 FEDEP Step 1

Identifying “operational objectives” is an important first step. Equally important is the next step to identify a definitive set of functional requirements associated with the conceptual model as illustrated in Figure 9-3.

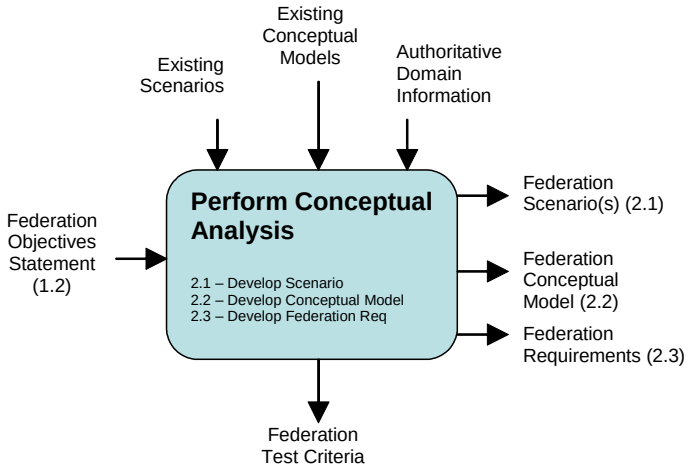

Figure 9-3 FEDEP Step 2

The operational objectives and functional requirements gathered and documented by following FEDEP steps 1 and 2 serves as the selection criteria for identifying available BOMs that match the need of the federation

under development. Essentially, the purpose and any conceptual model that is associated with a BOM can be analyzed and compared with the functional/behavior requirements of the anticipated composition. Conceivably, if no available BOMs match the selection criteria, which represents the intent or purpose of the conceptual model, then this information could be used to help generate a new BOM. Essentially, the selection criterion identifies an intended pattern of interoperability.

The collection of functional requirements, identified via the conceptual model, can be used to drive incremental and BOM-based construction of the federate or federation. This conceptual model driven development facilitates traceability and federation development automation for FEDEP steps 3 through 7 (see Figure 9-4).

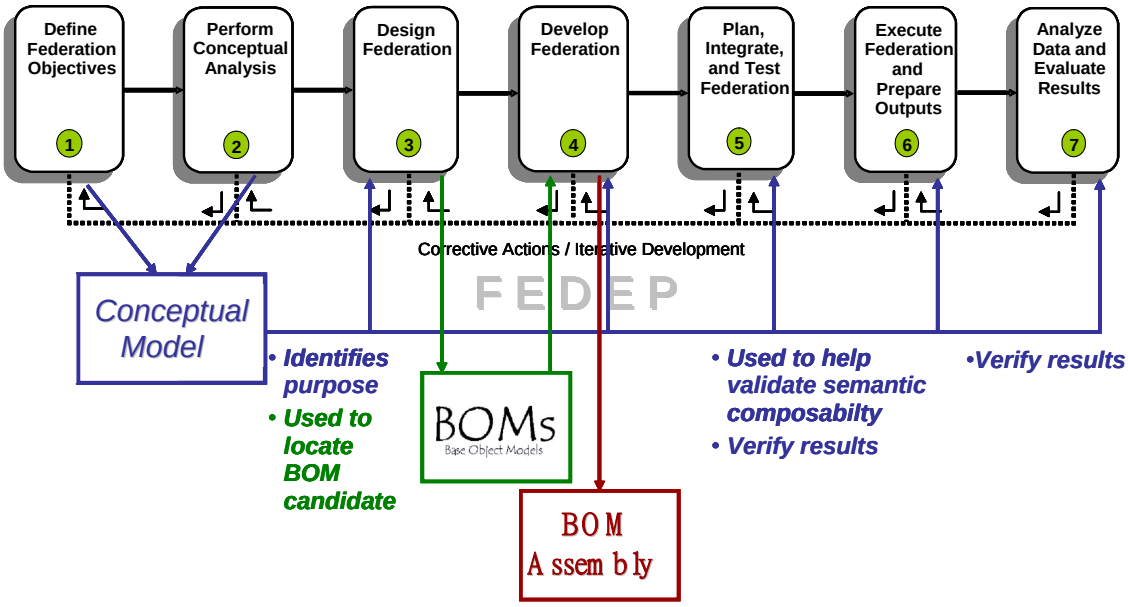

Figure 9-4 Conceptual Model Driven Development

9.3 The Identification of Candidate BOMs

Following the effort in establishing the functional requirements needed to represent a simulation or simulation space, the type of BOMs that are needed to fulfill overall objectives should be identified. This is representative of the efforts in step 3 of the FEDEP, which is illustrated in Figure 9-5.

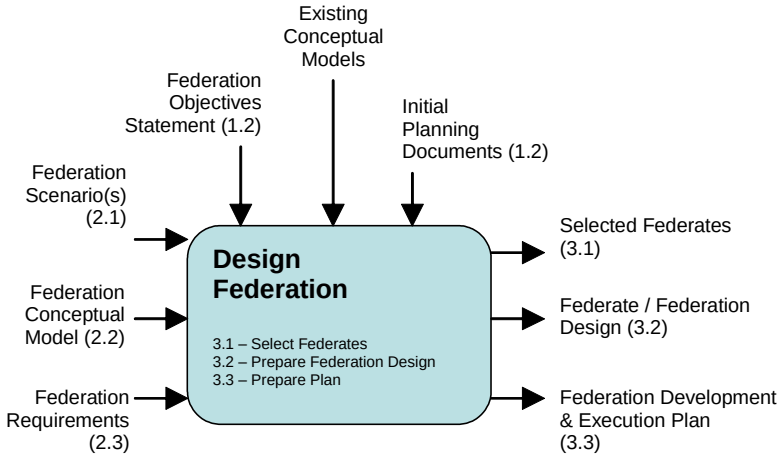

Figure 9-5 FEDEP Step 3

In establishing a BOM Assembly, the selection of federates may not be as important as recognizing what needs to be represented and reflected within the federation. As a starting point, use the operational objectives to determine the potential BOMs. This input information maps with FEDEP inputs 1.2, 2.1, 2.2, and 2.3.

The next step is to select (or develop, if necessary) appropriate BOMs and assemble them into a BOM Assembly. These activities are represented in step 4 of the FEDEP, which is shown in Figure 9-6.

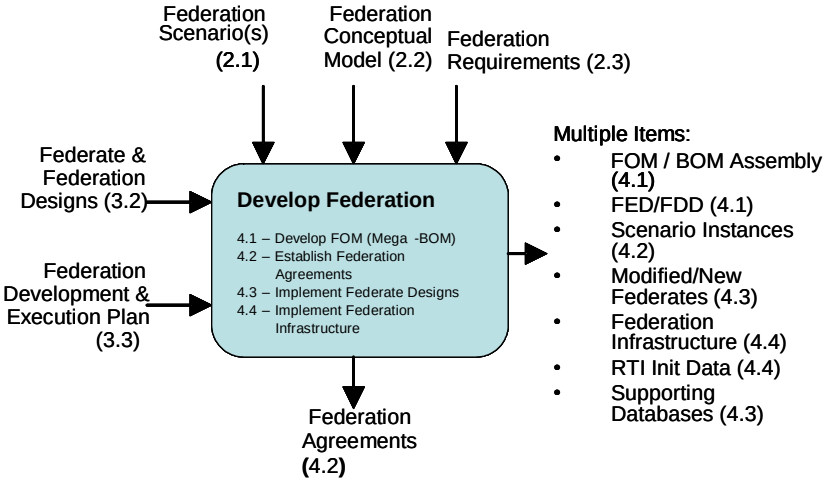

Figure 9-6 FEDEP Step 4

FEDEP step 4 suggests the use of existing BOMs, if they are available. During the design aspect of the Construction Stage (FEDEP steps 3 and 4), the federation engineer can use the functional requirement data to perform a “metadata match” with existing BOMs.

Theoretically, BOMs can be located simply by searching through each BOM’s metadata until a match is made; this is called metadata matching. Metadata matching, however, is always contingent on knowing the requirements for the BOM Assembly in construction.

If a BOM exists that helps to satisfy a requirement of the federate or federation being designed and developed (or updated), then it can be integrated into the BOM Assembly (see Figure 9-7). However if there is no satisfying BOM, then the opportunity exists to create a new BOM representing a *pattern of interplay* that supports the conceptual model or functional requirement. This new BOM can be immediately applied during design and development and saved for future reuse.

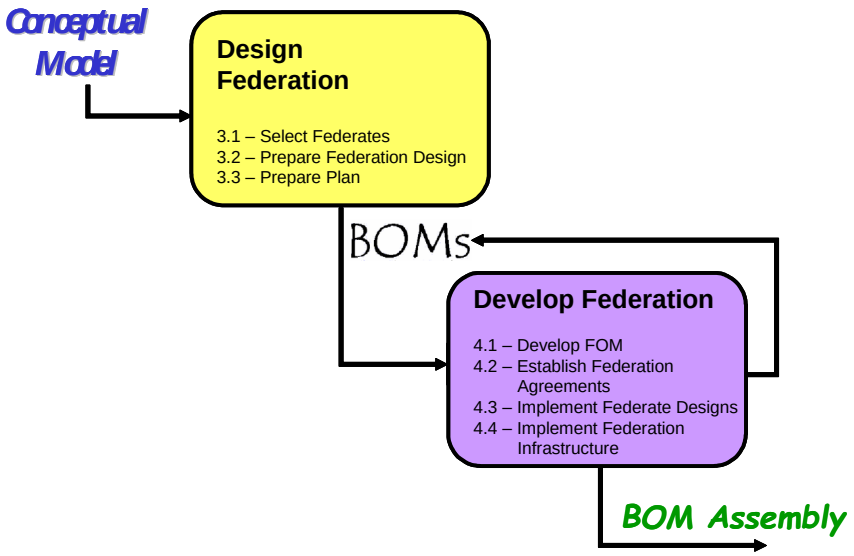

Figure 9-7 Base Object Model in FEDEP

9.4 The Assembly of BOMs

The most logical approach for integrating BOMs at the federation level is to create (or add to) a BOM Assembly in an incremental and iterative manner beginning with selecting BOMs that map with the requirements and conceptual models identified during conceptual analysis. This spiral development approach allows existing BOMs to be located that match with the framework element of a federate/federation under design. When unique threads (or *patterns of interplay*) to be represented by a federation or federation are identified from that skeleton design, the prospect of incorporating a particular BOM can be realized. If a candidate BOM does not exist, it creates an opportunity to build a BOM, which can then be shared and reused by others, as illustrated in Figure 9-7.

Leveraging the same template components as used for defining a BOM (see Table 9-1); a BOM Assembly can be composed using the *pattern of interplay* template component. Specifically, the use of one or more *patterns of interplay* is used to identify the higher-order pattern which the BOM Assembly is representing. As illustrated in Figure 9-8, associated with each *pattern of interplay* are one or more *pattern actions* (or *variants* to a *pattern action*) linked to a specific pattern BOM needed to carry out the activity. This approach also allows a BOM Assembly to be comprised of other lower level BOM Assemblies.

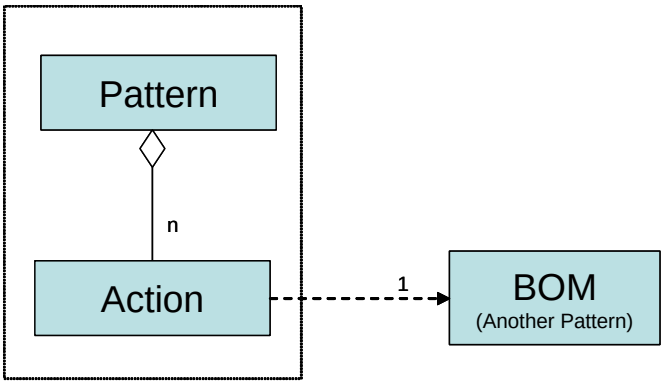

Figure 9-8 BOM Assembly – Pattern Action to BOM Relationship

As illustrated in Table 9-3, there are several ways a collection of BOMs can be woven into a BOM Assembly:

Table 9-3 BOM Assembly Approaches

BOM Assembly Approach	Description	When To Use
Multiple Patterns of Interplay	BOM Assembly is supported by multiple independent *patterns of interplay*	If the BOM Assembly is to represent a set of independent activities where the sequence of such activities is irrelevant and each activity is supported by a BOM, then the set of activities can be represented by multiple *patterns of interplay.* In this scenario, each BOM being assembled is associated with a single *pattern action* tied to one of multiple *patterns of interplay.*
Single Patterns of Interplay - Multiple Pattern Actions	BOM Assembly is supported by a single *pattern of interplay*	If the BOM Assembly is to represent a sequence of activities and each activity is supported by a BOM, then the activities can be represented by multiple *pattern actions* within a single *pattern of interplay.* In this scenario, each BOM being assembled is associated with one of multiple *pattern actions* tied to a single *pattern of interplay*.
Single Pattern of Interplay - Multiple Variants	BOM Assembly is supported by a single *pattern of interplay* containing multiple *variants*.	If the BOM Assembly is to represent a specific activity and there are multiple BOMs that may support that activity, then the activity can be represented by a *pattern action* within a single *pattern of interplay*. In this scenario, each BOM being assembled is associated with one of multiple *variants* of an individual *pattern action* tied to a single *pattern of interplay.*

Choosing which approach to use is dependent upon what is intended to be represented. However, any of these approaches can be mixed in achieving the creation of a BOM Assembly.

For further information regarding the integration of BOMs into a BOM Assembly, see the relevant papers referenced in the Table 2-2.

9.5 HLA Compatibility Support

While the use of BOMs does not mandate the use of HLA, it is highly likely that BOM Assemblies produced during the federation development process are intended to support an HLA based federation. A transformation from a BOM Assembly to a FOM is quite possible by adhering to the HLA OMT. It should be noted that the

generated FOM will contain less *metadata* than the BOM Assembly source because the *model identification* structure is different.

The additional *metadata* carried in a BOM is intended to make it easier for object-oriented development tools and simulation software designers to discover and use applicable BOMs for the assembly of HLA FOMs, SOMs, and BOM Assemblies, and in assembling simulation spaces.

For further information regarding the transformation of a BOM Assembly into an HLA FOM, see the relevant papers referenced in the Table 2-2.

9.6 Use History and Feedback (Metadata)

Part of making a BOM useful to others is hinged upon reflecting the integration experience others have had pertaining to the BOM. It is imperative that as BOMs are used and applied in the construction or modification of a BOM Assembly, the integration experience regarding a BOM's use (and success) is shared with the communities at large who either use or develop BOMs. This feedback involves sharing the integration experience in the form of BOM *metadata* as defined in the BOM Template Specification. This integration experience information can be reflected in the *use history* field of the BOM Model Identification *metadata*. Much like the *purpose metadata* element, the *use history metadata* element allows BOMs to be more easily matched and used for future development.

A likely mechanism for canvassing integration experience as *use history metadata* is to provide an online collection mechanism affiliated with a reuse library where the BOM was retrieved. The idea of reflecting *use history* is somewhat analogous to the reader comments and reviews that web sites such as Amazon.com provide for potential customers of books, records, and videos.

This page left intentionally blank

10 BOM Distribution

The most practical way to encourage the reuse of BOMs is to provide a mechanism for distribution. The following SISO BOM Standardization Pipeline has been identified to assist in BOM distribution.

The SISO BOM Component Standardization Pipeline is composed of the following steps:

1. Create a BOM
2. Test and Verify BOM (use it)
3. Submit BOM (with metadata) to a compliant resource reuse library.
4. Make BOM available to user-base via web interface/service
5. Encourage feedback of integration via “Use history” metadata.

10.1 Tools

Finding HLA OMT tools that know how to reflect a BOM so that it can be either built or used is a current deficiency that the BOM PDG anticipates will also be resolved in the near future. Tools should support the interface aspect of a BOM even though it may not provide the appropriate metadata and XML behavior modeling representation support. Over time, tools should evolve to support these capabilities.

10.2 Resource Reuse Library

As shown in the pipeline previous identified in Figure 9-1, a reuse library can be used to provide publicly available BOMs and BCIs developed by various simulation software developers and organizations. Such a notional reuse library and its application for creating compositions for the interface view of federates and federations is depicted in Figure 10-1.

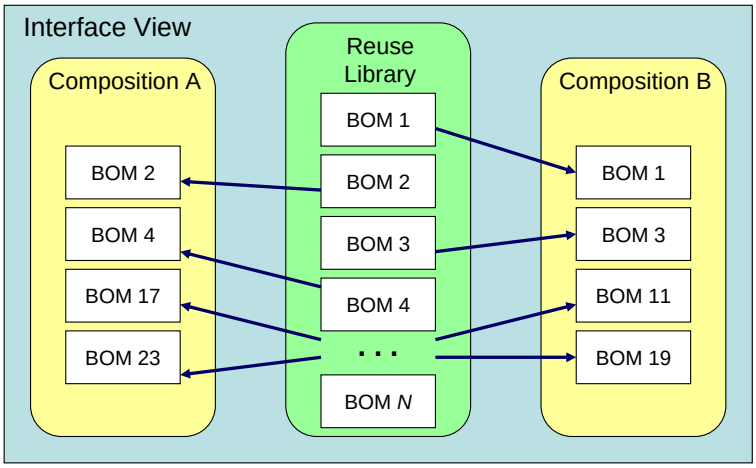

Figure 10-1- Leveraging a BOM Resource Reuse Library

Currently, libraries that can support BOMs and BCIs are not available, which raises the issue as to where or when BOMs can be submitted and obtained from an online reuse library. A reuse library will need to provide the following three capabilities:

- A mechanism to upload and download BOMs
- A mechanism to add integration use history (as metadata) to available BOMs within the reuse library in order to encourage further use of BOMs
- A search engine based on the requirements and use history metadata

As BOMs are used and exercised by various federates, integration use history metadata should also be appended and updated accordingly for the BOM (as a reference) and provided back to the reuse library where the BOM was obtained.

The search engine would serve as a useful mechanism for federation engineers to match and find applicable BOMs and BCIs for their federation development or maintenance effort (based on their requirements).

This page left intentionally blank