Simulation Interoperability Standards Organization (SISO)

Base Object Model (BOM) Template Specification

SISO-STD-003-2006

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 (BOM) 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

Bob Lutz
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 7
1.1 PURPOSE 7
1.2 SCOPE 7
1.3 OBJECTIVE 8
1.4 INTENDED AUDIENCE 9
2 REFERENCES 11
3 DEFINITIONS 13
4 ACRONYMS AND ABBREVIATIONS 15
5 CONVENTIONS 17
5.1 NAMES 17
5.2 DOT NOTATION 17
6 BOM TEMPLATE COMPONENTS 19
6.1 MODEL IDENTIFICATION 23
6.1.1 Purpose/Background 23
6.1.2 Table Format 23
6.1.3 Inclusion Criteria 26
6.1.4 Example 26
6.2 CONCEPTUAL MODEL DEFINITION 28
6.2.1 Pattern of Interplay 28
6.2.2 State Machine 31
6.2.3 Entity Type 34
6.2.4 Event Type Definition 35
6.3 MODEL MAPPING 41
6.3.1 Entity Type Mapping 41
6.3.2 Event Type Mapping 44
6.4 OBJECT MODEL DEFINITION 49
6.5 SUPPORTING TABLES 51
6.5.1 BOM Notes 52
6.5.2 BOM Lexicon Definitions 53
7 BOM DIF SCHEMA 59
7.1 MODEL IDENTIFICATION 60
7.2 CONCEPTUAL MODEL DEFINITION 63
7.2.1 Pattern of Interplay 63
7.2.2 State Machine 65
7.2.3 BOM Entity Types 66
7.2.4 BOM Event Types 67
7.3 MODEL MAPPING 68
7.3.1 Entity Type Mappings 68
7.3.2 Event Type Mappings 68
7.4 OBJECT MODEL DEFINITION 69
7.5 NOTES 71
ANNEX A – BOM SCHEMAS 73
7.6 BOM SCHEMA 73
7.7 MODEL IDENTIFICATION SCHEMA 79
7.8 HLA OMT SCHEMA 85
ANNEX B – BOM EXAMPLE 103
ANNEX C – XMLSPY® SCHEMA DESIGN CONTENT MODEL 115

This page left intentionally blank

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 of interplay* is captured in the BOM.

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

1.1 Purpose

The “Base Object Model (BOM) Template Specification” defines the format and syntax for describing the elements of a template for representing BOMs. It specifies the semantics of the elements of a BOM and the syntax for constructing BOMs from those elements. In addition, this specification provides a data interchange format (DIF) for the representation of BOMs using the Extensible Markup Language (XML). The BOM DIF enables tools to exchange and reason about BOMs.

1.2 Scope

The “Base Object Model (BOM) Template Specification” defines the semantics and the syntax needed to represent a BOM. 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.

Specifically, 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 described in terms of class structures that collectively define the inherit capabilities of a simulation application. For a BOM, this interface description, identified as the Object Model Definition, is defined using 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.

This document characterizes these essential elements by specifying a basic BOM Template for documenting information needed to identify and assess the reusability of BOMs.

"The Base Object Model (BOM) Guidance Specification" provides discussion on BOM development and the application and use of BOMs for the assembly of simulations and simulation environments as illustrated in Figure 1-1.

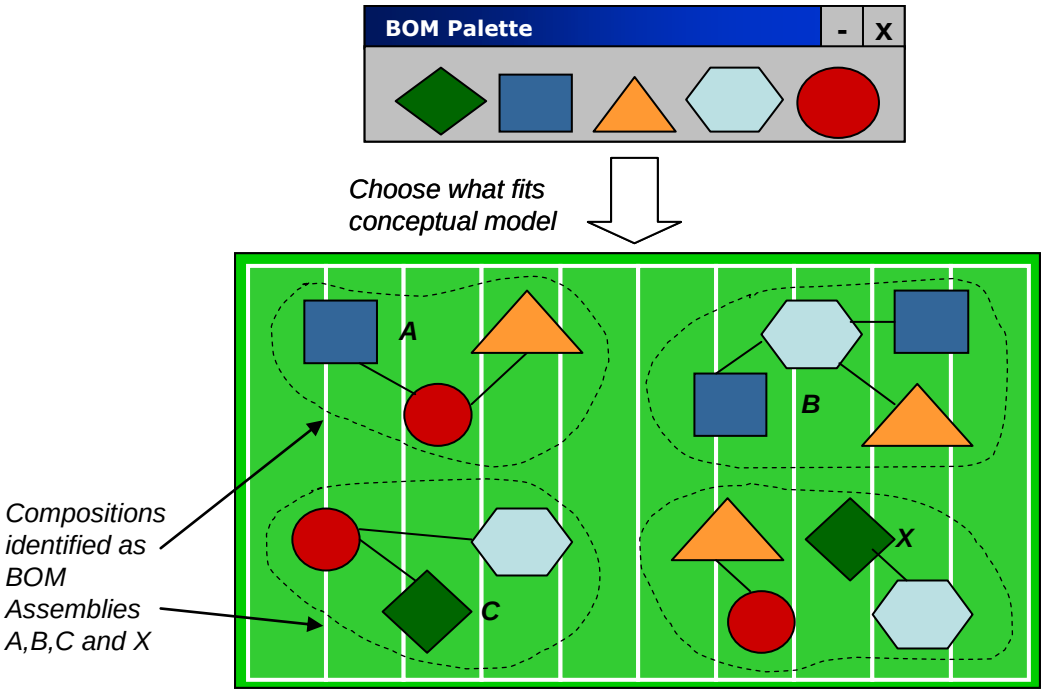


Figure 1-1 BOM Composability View

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 defines the template components for capturing the information and the XML schema for interchanging the information.

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 specification 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-003.0	Guide for Base Object Model (BOM) Use and Implementation
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, 7th Edition
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/

* 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 these terms in the scope of this document.

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

BOM	Base Object Model
BCI	BOM Component Implementation
BNF	Backus-Naur Form
COI	Community of Interest
DIF	Data Interchange Format
DoD	Department of Defense
EXCOM	Executive Committee
FDD	Federation Object Model (FOM) Document Data
FEDEP	Federation Development and Execution Process
FOM	Federation Object Model
HLA	High Level Architecture
IEEE	Institute of Electrical and Electronics Engineers, Inc.
M&S	Modeling and Simulation
OMT	Object Model Template
PDG	Product Development Group
POC	Point of Contact
RFOM	Reference Federation Object Model
RTI	Runtime Infrastructure
SAC	Standards Activity Committee
SISO	Simulation Interoperability Standards Organization
SOM	Simulation Object Model
SRML	Simulation Reference Markup Language
URI	Uniform Resource Identifiers
XML	Extensible Markup Language
XSD	XML Schema Definition

This page left intentionally blank

5 Conventions

Conventions in this document specify how entries are to be formulated when completing a BOM. The following conventions pertain to this specification.

5.1 Names

Names in object models shall adhere to XML naming conventions. XML conventions require that names be constructed from a combination of letters, digits, hyphens, colons, full stops (periods), and underscores with no spaces or other breaking characters (e.g., tabs, carriage returns, etc.). For object model definition elements, which are described using HLA OMT constructs, allowable names shall be further restricted as follows:

a) The period (full stop) is reserved for qualifying class names and shall not be included in a user-defined name within an object model. When fully qualified, the class name includes all predecessor (i.e., superclass) class names (separated by periods) beginning at the root and ending at the class name in question.
b) As recommended in the XML specification, the colon character shall not be used.
c) Names beginning with the string “hla,” or any string that would match (('H'|'h') ('L'|'l') ('A'|'a')), are reserved and shall not be included in user-defined names.
d) The case of all textual data within BOMs (including names) shall be significant; all textual data defined in an object model document shall be case-sensitive.
e) A name consisting of the string “na,” or any string that would match (('N'|'n') ('A'|'a')), is reserved to indicate that a name is not applicable in this circumstance and shall not be included as a user-defined name.

These rules apply to the following names in BOMs:

- Model Identification
- Pattern of Interplay
- Pattern Action
- State Machine
- State
- Entity Type
- Event Type
- HLA Object Class
- HLA Interaction Class
- HLA Attribute
- HLA Parameter
- Datatype
- Enumerated Datatype
- Fixed Record Field
- Variant Record Alternative
- Basic Data Representation
- Note

5.2 Dot Notation

Dot Notation is a text convention for describing the affiliation of items, such as object classes and attributes, by placing them within a hierarchical scheme that is representative of a tree structure. Dot Notation reads left to right or from the root of the tree out to branches and leaves. Periods separate the levels or branches. A Dot Notation is used to ensure that the reference within the template component is unambiguous.

For example, the scheme **Platform.land.tank.M1.turret** is used to identify and use an **M1** tank as the *HLA object class* of interest, and the **turret** as the *HLA attribute* that is of interest. Both the **M1** and **turret** are easily referenced by the class hierarchy tree defined in the text.

Although individual names need not be unique, all *entity types*, *event types*, *pattern actions*, *HLA object classes*, and *HLA interaction classes* within a BOM shall be uniquely identifiable when concatenated (via Dot Notation) with the names of higher level superclasses. If a *pattern action*, *characteristic*, *HLA attribute*, or *HLA parameter* is used to identify a component in the BOM, its class hierarchy shall be specified in Dot Notation to the extent necessary to identify the action, characteristic, attribute, or parameter uniquely.

6 BOM Template Components

A BOM is composed of a group of interrelated elements specifying metadata information, conceptual model information, class structure information defined using HLA OMT constructs, and the mapping between conceptual model elements and object model elements that identify the class structure information. Figure 6-1 provides an illustration of the BOM template, which is made of four major template component views: Model Identification, Conceptual Model Definition, Model Mapping, and Object Model Definition. In addition, Notes and Lexicons can be provided to clarify the semantics of a BOM.

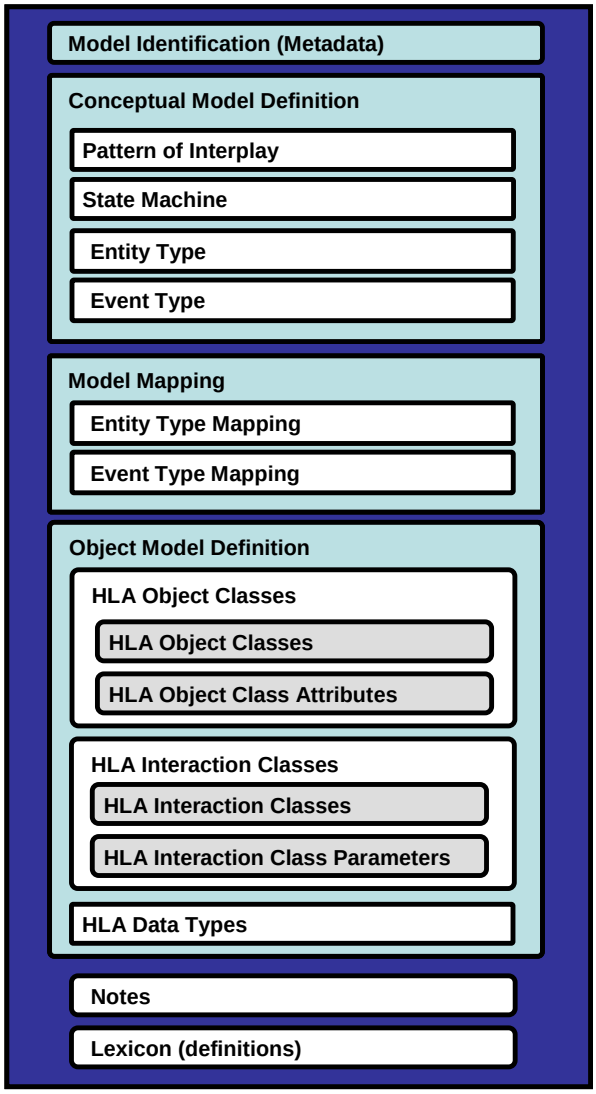


Figure 6-1 BOM Composition

The BOM template components provide mechanisms for defining Conceptual Model Definition elements such as *patterns of interplay*, *state machines*, *entity types,* and *event types*; defining Object Model Definition elements such as class structures and data types; and identifying Model Mappings between *entity types* and *event types* contained within a Conceptual Model Definition and class structures contained within an Object Model Definition.

The information content of these template components can be represented in many different ways or presentations. A presentation is the formatting of the information contained in the BOM in a particular manner for a particular purpose. For example, the combination of tabular and graphical formats presented in this section is designed for presentation on a printed page, whereas the BOM DIF (as defined in Section 7) is a presentation designed for passing a BOM between tools. All presentations shall be compatible with the information content of the BOM DIF. By meeting that requirement, it follows that the BOMs can be represented in the tabular and graphical formats demonstrated in the following sections. It is the necessary information content of the BOM and BOM DIF that are standardized in this specification. The tabular and graphical depictions of the information are provided to illustrate the semantic concepts and syntactical relationships. While these may be used in representing BOMs, users are free to provide the same information in alternative presentation formats.

The BOM template consists of a set of template components based upon the original IEEE Std 1516.2-2000 HLA OMT Specification, which hereinafter will be referred to as the HLA OMT Specification, and additional template components defined in this document. The full set of template components used for representing BOMs is identified in Table 6-1. The template components leveraged from the HLA OMT Specification are not redefined here. The application of the template components in the list below differs from that in the HLA OMT Specification in that they are used for specifying aspects of a conceptual model, simulation object model, and/or federation object model, which can be used and reused in the design, development, and extension of interoperable simulations.

Table 6-1 BOM Template Components

Category	Template Component	Description
Model Identification	Model Identification (Metadata)	Associates the important identifying information (i.e., metadata) with the BOM. This is an extension of the Object Model Identification Table found in the HLA OMT Specification.
Conceptual Model Definition	Pattern of Interplay	Identifies the *patterns of interplay* including the *pattern actions*, *variations,* and *exceptions* required for representing the activities associated with each *pattern of interplay*.
	State Machine	Identifies the *state machines* including the identification of *conceptual entities* and *states* required for representing the aspects of the conceptual model.
	Entity Type	Identifies the types of *conceptual entities* required for representing the aspects of the conceptual model.
	Event Type	Identifies the types of *conceptual events*, whether directed (*messages*) or undirected events (*triggers*), required for representing the *pattern actions* associated with a *pattern of interplay*.
Model Mapping	Entity Type Mapping	Identifies and associates what class structures, using HLA OMT Specification constructs, can be used to represent the *entity types* identified in the Conceptual Model Definition of a BOM.
	Event Type Mapping	Identifies and associates what class structures, using HLA OMT Specification constructs, can be used to represent the *event types* identified in the Conceptual Model Definition of a BOM.
Object Model Definition	Object Class Structure	Records the list of *HLA object classes* related to *patterns of interplay* and and describes their class-subclass relationships. This structure is unchanged from the HLA OMT Specification.
	Interaction Class Structure	Records the list of *HLA interaction classes* related to *patterns of interplay* and and describes their class-subclass relationships. This structure is unchanged from the HLA OMT Specification.
	Attribute	Specifies features of *HLA object class attributes* in a *pattern of interplay*. This structure is unchanged from the HLA OMT Specification.
	Parameter	Specifies features of *HLA interaction class parameters* in a *pattern of interplay*. This structure is unchanged from the HLA OMT Specification.

	Datatype	Specifies details of data representation in the object model. This structure is unchanged from the HLA OMT Specification.
Supporting Tables	Notes	Expands explanations of any BOM DIF element. This structure is unchanged from the HLA OMT Specification.
	Lexicon Definition	Defines all of the Conceptual Model Definitions (*patterns of interplay*, *state machines*, *entity types,* and *event types*), and Object Model Definitions (*HLA object classes* and their *attributes*, *HLA interaction classes* and their *parameters*) used in the BOM.

The views of the Conceptual Model Definition follow the structure of the template components defined exclusively within this specification. These template components include *patterns of interplay*, *state machines*, *entity types,* and *event types*. The views of the Model Mapping also follow the structure of the template components defined exclusively within this specification. These template components include *entity type mappings* and *event type mappings*. The views of the Object Model Definition, however, follow the same structure in a BOM as they do in an HLA FOM or SOM as defined in the HLA OMT Specification. These template components include *HLA object classes*, *HLA interaction classes*, *HLA attributes*, and *HLA parameters*. These Object Model Definition template components provide a mechanism to expose the class structures that can be represented by federation participants thereby defining an interface, which can be mapped to views of a Conceptual Model Definition. The views of the Supporting Tables also follow the structure of the template components defined in the HLA OMT Specification. These template components include *notes* and *lexicon definitions*. The last BOM template component, *lexicon definition*, is essential to ensure that the semantics of the terms used in a BOM are understood and documented.

Figure 6-2 identifies the collection of the template components used to define a BOM.

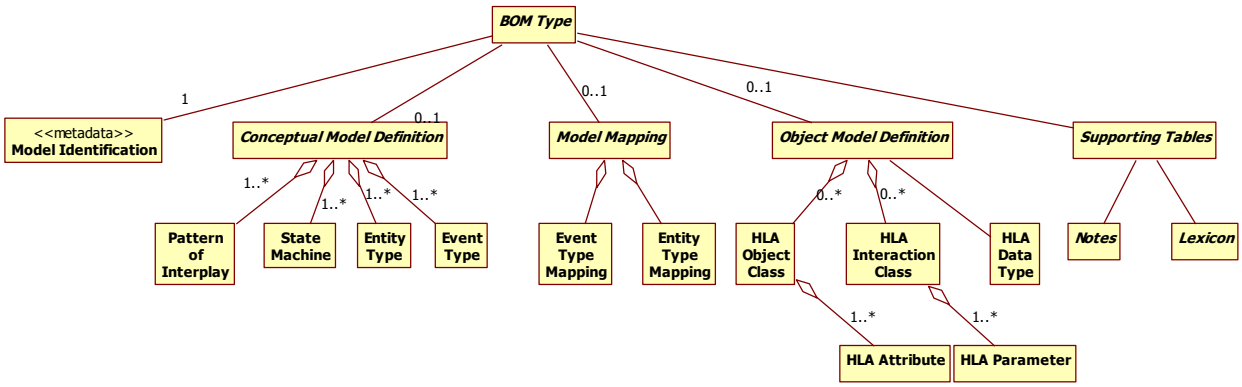


Figure 6-2 BOM Template Components

The basics of each BOM template component are presented in the following separate sub-clauses. For each template component, a unique *table format* is provided and described with criteria suggested to help guide decisions on completing specific categories within each of these template components.

The BOM *table formats* presented in Tables 6-2, 6-5, 6-8, 6-11, 6-14, 6-17, 6-20, 6-29, 6-31, 6-33, 6-35, 6-37, 6-39, 6-41, and 6-43 use a subset of Backus-Naur Form (BNF) to specify the information content that belongs in particular table cells. In BNF, the type of information to be substituted in the table is enclosed in angle brackets (e.g., <description>).[5]

[5] Naur, P. et al., "Report on the Algorithmic Language ALGOL 60," Communications of the ACM, vol. 6, no. 1, pp. 1-17, January 1963.

Unless noted otherwise in the "Guide for Base Object Model (BOM) Use and Implementation," all of the BOM template components shall be completed when specifying a BOM. However certain tables may be empty or devoid of specific content. Fields that support optional information, but have no value for a specific table instance, shall be filled with “na.” The specific rules for the applicability of each BOM template component are provided in the *information description table*, which follows the *table format*.

Unless otherwise stated in the *information description table*, each row within a table shall appear only once. A comma separated list shall be used to identify multiple items in a table cell (i.e., *sender*, *receiver,* and anything at the lowest level that has been identified in the *information description table* as “many”). A *notes* entry may be identified for any table element to provide amplifying information.

Example use of each template component table is also provided; these examples are for illustrative purposes only and are not meant to imply any additional requirements. Although *notes* are not explicitly included in the *table format* or identified in the *template format description*, they can be included in the tables as illustrated in the examples. Examples of *lexicon definitions* have been provided as well.

Any BOM that fully conforms to all of the rules and constraints stated in this specification shall be considered a compliant Base Object Model.

6.1 Model Identification

6.1.1 Purpose/Background

A design goal for all BOMs is to facilitate reuse. BOMs provide information that enables inferences to be drawn regarding their reuse potential for supporting the extension and creation of models, simulations, and federations. It is important to include a minimum but sufficient degree of descriptive information in the BOM. For instance, when federation developers wish to pose detailed questions to those who were responsible in the development and distribution of the BOM, point-of-contact (*POC*) information within a BOM is important. The purpose of the Model Identification is to document certain key *metadata* information about the BOM.

6.1.2 Table Format

The Model Identification Table is the template component of the BOM DIF used to identify the basic *metadata* of a BOM and is provided in Table 6-2.

Table 6-2 Model Identification Table Format

Category	Information
Name	<name>
Type	<type>
Version	<version>
Modification Date	<date>
Security Classification	<security classification>
Release Restriction	<release restriction>
Purpose	<purpose>
Application Domain	<application domain>
Description	<description>
Use Limitation	<limitation>
Use History	<history>
Keyword	
Taxonomy	<taxonomy>
Keyword Value	<keyword value>
POC	
POC Type	<poc type>
POC Name	<poc name>
POC Organization	<poc organization>
POC Telephone	<poc telephone>
POC Email	<poc email>

Reference	
Type	<ref type>
Identification	<identification>
Other	<other>
Glyph	<image>
Type	<type>
Alternate	<alt>
Height	<height>
Width	<width>

- The first column (Category) specifies the categories of data that shall be provided in the table.
- The second column (Information) specifies the required information that shall be identified in the table.

Table 6-3 provides a description of the categories of information pertaining to the Model Identification *metadata*. Many of the information categories used for the Model Identification view are leveraged from the HLA OMT Specification, however there are some additional information categories that were added, which are noted in the IEEE Std 1516.2-2000 column of Table 6-3. Italics are used in the Values column of Table 6-3 to denote the type of data that shall be provided (e.g., text). Normal font is used in this column to denote potential literal values.

Table 6-3 Model Identification Information Description Table

Category	Description	Occurs	Values
Name	This field shall specify the name assigned to the object model.	1	*text*
Type	This field shall specify the type that the object model represents; for a BOM the only valid value is "BOM."	1	BOM
Version	This field shall specify the version identification assigned to the object model.	1	*text*
Modification Date	This field shall specify the latest date on which this version of the object model was last modified. If the model is still in its original version (has not undergone modification) this field shall contain the creation date. The modification date shall be specified in the format "yyyy-mm-dd" (e.g., 1999-04-15).	1	*yyyy-mm-dd*
Security Classification	This field shall contain the security classification of the object model.	1	Unclassified, Confidential, Secret, Top Secret, *other text*
Release Restriction	This field shall contain any restrictions on the release of the object models to specific organizations or individuals. Multiple rows are permissible if multiple release restrictions exist.	0..many	*text*
Purpose	This field shall specify the purpose for which the BOM was developed.	0..1	*text*
Application Domain	This field shall specify the type or class of application to which the BOM pertains.	0..1	Analysis, Training, Test and Evaluation, Engineering, Acquisition, *other text*
Description	This field shall provide an account of the content of the BOM.	1	*text*
Use Limitation	This field shall provide any known applications for which this BOM has been found not to be appropriate.	0..1	*text*
Use History	This field shall provide a description of where this BOM has been used in the construction of other object models.	0..many	*text*
Keyword	This pair of fields shall be repeated as many times as necessary.	0..many	
Taxonomy	This field specifies the source of the keyword vocabulary.	0..1	*text*
Keyword Value	This field shall provide the word or concept that is addressed by the BOM.	1	*text*

POC		This set of fields shall specify an organization or a person who has a particular role with respect to the BOM. At least one set of POC information shall be supplied. Multiple sets may be supplied.	1..many	
	POC Type	This field shall specify the role that the POC has with respect to the BOM.	1	Primary author, Contributor, Proponent, Sponsor, Release Authority, Technical POC, *other text*
	POC Name	This field shall specify the name of the POC, including an honorific (e.g., Dr., Ms., etc.) or rank, first name, and last name where appropriate. In the case where the POC is an organization, this field is optional, but either a POC Name and/or a POC Organization shall be specified.	0..1	*Text*
	POC Organization	This field shall specify the organization if the POC is generalized to an organization. If a POC Name is specified, then this field is optional, and contains the name of the organization with which the person is affiliated.	0..1	*Text*
	POC Telephone	This field shall specify the telephone number for the POC including the international telephone code for the POC's country.	0..many	*Text*
	POC Email	This field shall specify the email address of the POC.	1..many	*text*
Reference		This set of fields shall specify a pointer to additional sources of information.	0..many	
	Type	This field shall specify the way in which the reference is related to the BOM.	1	Source material, Conceptual model, Related BOM, *other text*
	Identification	This field shall specify how to locate the reference source. Examples include a Uniform Resource Identifier (URI), XML reference ID (ref ID), or ISBN.	1	*Text*
Other		This field shall specify other data deemed relevant by the author of the object model.	0..1	*Text*
Glyph		This field holds the image, which can be used to visually represent a BOM in a tool palette or web-based repository.	0..1	*base64*
	Type	This field holds the image type being represented.	1	BMP, GIF, JPG, PNG, TIFF, *other text*
	Alt	This field shall be used to provide alternative text in case the image represented in the *Image* field cannot be displayed.	0..1	*Text*
	Height	This field shall specify the pixel height of the glyph image represented in the *Image* field.	0..1	*Short*
	Width	This field shall specify the pixel width of the glyph image represented in the *Image* field.	0..1	*Short*

The relationship of the Model Identification *metadata* sub-elements within a BOM is illustrated in Figure 6-3.

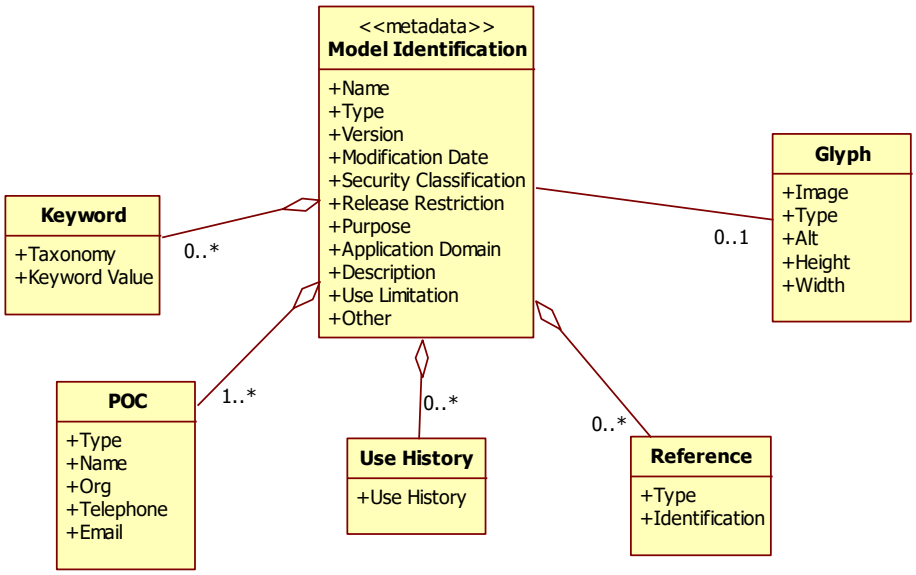


Figure 6-3 BOM Model Identification Template Component Elements

Figure 6-3 illustrates that a *model identification* may consist of multiple *keywords*, multiple *POCs*, multiple *use histories*, multiple *references,* and a *glyph* image.

6.1.3 Inclusion Criteria

Every BOM shall have Model Identification *metadata*. The categories of information specified in Table 6-2 shall be included for all Model Identification *metadata* elements unless “0..1” or “0..many” is identified in the Occurs column of Table 6-3.

6.1.4 Example

Table 6-4 illustrates a simple example of the Model Identification Table.

Table 6-4 Model Identification Table Example

Category	Information
Name	Restaurant Payment
Type	BOM
Version	1.0 Beta
Modification Date	2004-05-04
Security Classification	Unclassified
Release Restriction	Not for release outside the XYZ Restaurant Corporation.
Release Restriction	Release only to BOM PDG members.
Purpose	Standardize the pattern of interplay between Customer and Waiter for payment of meal at restaurant.
Application Domain	Restaurant operations
Description	This is an example to illustrate the key concepts of a BOM.
Use Limitation	Not applicable to drive-through restaurants.
Use History	Used as example in the BOM Template Specification
Keyword	
Taxonomy	Restaurant taxonomy
Keyword Value	Restaurant
Keyword	
Taxonomy	Commerce taxonomy
Keyword Value	Payment
POC	
POC Type	Primary author
POC Name	Mr. Snuffy Smith
POC Organization	XYZ Restaurant Corporation
POC Telephone	+1 44 123-456-7890
POC Email	snuffy.smith@goodeatscafe.com
POC	
POC Type	Release authority
POC Name	Na
POC Organization	XYZ Restaurant Corporation
POC Telephone	+1 44 123-456-1000
POC Email	the.lawyer@xyz-foods.com
Reference	
Type	Glossary
Identification	ISBN 12345678901
Reference	
Type	Conceptual Model
Identification	http://boms.info/resaurantconceptualmodel.doc
Other	This BOM pattern of interplay was featured on the Food Network special “HLA and the Future of Restaurant Operations”
Glyph	
Type	JPG
Alt	Payment
Height	32
Width	32
Note	*[1]

6.2 Conceptual Model Definition

Since the BOM can be used to represent a conceptual model, additional information is needed to document how the *pattern of interplay* within the conceptual model takes place, the various *state machines* that might be represented, and the simulation conceptual elements identified as *entity types* and *event types* that are used. This section describes the various mechanisms for capturing the elements of a Conceptual Model Definition that can be represented by a BOM.

6.2.1 Pattern of Interplay

6.2.1.1 Purpose/Background

The *pattern of interplay* template component provides a mechanism for identifying sequences of *pattern actions* (including *variations* and *exceptions*) necessary for fulfilling a *pattern of interplay*, which may be represented by a BOM.

The activities for a *pattern action* can be represented by either a defined *event type* within the BOM (see Section 6.2.4), or by another BOM, which provides greater detail of the activities necessary for fulfilling similar types of actions. The *pattern of interplay* template component also supports the ability to identify *exceptions* and *variations* that can be associated with a *pattern action*. An *exception* is an alternative action that may occur that typically causes the remaining sequences of the *pattern of interplay* to fail (where the next *pattern action* in the sequence may not be performed). A *variation*, however, identifies a different way for a *pattern action* to be accomplished without impeding the completion and success of the *pattern of interplay*. The *pattern actions* of a *pattern of interplay* may be used by a *state machine* (see Section 6.2.2) to identify the *exit condition* required to cause a transition from one state to another.

6.2.1.2 Table Format

The Pattern of Interplay Table is the template component of the BOM DIF used to identify *patterns of interplay* at the Conceptual Model Definition level and is provided in Table 6-5.

Table 6-5 Pattern of Interplay Table Format

Pattern of Interplay Name		**Sequence**	**Name**	**Sender**	**Receiver**	**Event**	**BOM**	**Condition**
<pattern of interplay name>	**Pattern Action**	<sequence>	<pattern action name>	<sender name>	<receiver name>	<event name>	<BOM name>	
	Exception		<exception name>	<sender name>	<receiver name>	<event name>	<BOM name>	<condition>
	Variation		<variation name>	<sender name>	<receiver name>	<event name>	<BOM name>	<condition>

One or more *patterns of interplay* can be identified by this table. As depicted in Table 6-6, each *pattern of interplay* can define one or more *pattern action(s)* including *exceptions* and *variations*; the types of *conceptual entities* involved in sending and/or receiving a *pattern action*; and the BOM *event types* or other *BOMs* used for fulfilling the activities of a *pattern action*.

Table 6-6 Pattern of Interplay Information Description Table

Category	Description	Occurs	Values
Pattern of Interplay	Identifies one or more *patterns of interplay*	1..many	
Name	Identifies the name of the *pattern of interplay*	1	*text*
Pattern Action	Identifies a *pattern action* to be carried out in order to successfully accomplish the named *pattern of interplay*	1..many	
Sequence	Identifies the sequence order of the main *pattern action*	1	*text*
Name	Identifies the name of the main *pattern action*	1	*text*
Sender	Identifies the conceptual *entity type(s)* responsible for initiating the *pattern action*	1..many	*text*
Receiver	Identifies the conceptual *entity type(s)* intended to be the recipient of the *pattern action*	1..many	*text*
Event	Identifies the *event type* that represents the activity associated with this *pattern action*	0..1	*eventIdentifer*
BOM	Identifies another BOM that represents the activity associated with this *pattern action*	0..1	*bomIdentifier*
Exception	Identifies undesired but potential behavior for the *pattern action*	0..many	
Name	Identifies the name of the *exception*	1	*text*
Sender	Identifies the conceptual entity type(s) responsible for initiating the *pattern action* (event or BOM) representative of the *exception*	1..many	*text*
Receiver	Identifies the conceptual entity type(s) intended to receive the action (event or BOM) representative of the *exception*	0..many	*text*
Event	Identifies the *event type* that represents the activity associated with this *exception*	0..1	*eventIdentifer*
BOM	Identifies another BOM that represents the activity associated with this *exception*	0..1	*bomIdentifier*
Condition	Identifies the condition(s) required for the *exception* to occur	1..many	*text*
Variation	Identifies a different way the *pattern action* can be successfully accomplished	0..many	
Name	Identifies the name of the *variation*	1	*text*
Sender	Identifies the conceptual entity type(s) responsible for sending the action (event or BOM) representative of the *variation*	1..many	*text*
Receiver	Identifies the conceptual entity type(s) intended to be the recipient of the action (event or BOM) representative of the *variation*	1..many	*text*
Event	Identifies the *event type* that represents the activity associated with this *variation*	0..1	*eventIdentifer*
BOM	Identifies another BOM that represents the activity associated with this *variation*	0..1	*bomIdentifier*
Condition	Identifies the condition(s) required for the *variation* to occur	0..many	*text*

The relationship of the *pattern of interplay* sub-elements within a BOM is illustrated in Figure 6-4.

Figure 6-4 illustrates that a *pattern of interplay* may consist of multiple *pattern actions,* and each *pattern action* may consist of multiple *variations* and *exceptions.* Furthermore, a *pattern action, variation,* or *exception* consists of one or more *senders* and *receivers,* and either an *event* or *BOM*. A *sender* or *receiver* uses a specific BOM *entity type*, and an *event* uses a specific BOM *event type*, whereas a *BOM* uses another *BOM type*. Section 6.2.3 describes the template components needed for defining BOM *entity types*. Section 6.2.4 describes the template components needed for identifying BOM *event types* (as either *triggers* or *messages*).

A Dot Notation shall be used for the *pattern of interplay* values as needed to ensure that the reference within the template component is unambiguous. Thus, *sender*, *receiver, event,* and *BOM* names associated to a *pattern action*, *variation,* or *exception* shall use Dot Notation as necessary, which may include the full

parentage to uniquely identify the *entity type* (i.e., *sender*, *receiver*), *event type* (i.e., *event*) or *bom type* (i.e., *BOM*) being used.

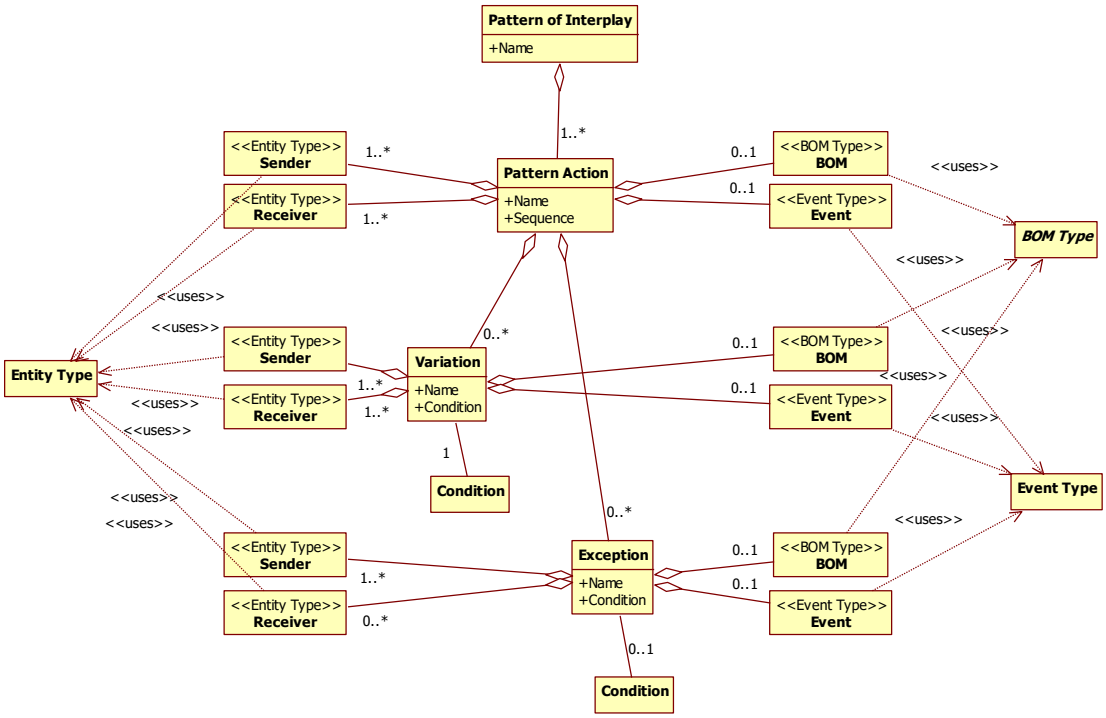


Figure 6-4 BOM Pattern Action Relationship

6.2.1.3 Inclusion Criteria

Every BOM that includes the Conceptual Model Definition section shall either contain a *pattern of interplay* template component or reference elements of a *pattern of interplay* defined within another BOM. The categories of information specified in Table 6-5 shall be included for all *pattern of interplay* elements unless "0..1" or "0..many" is identified in the Occurs column of Table 6-6.

6.2.1.4 Example

Table 6-7 provides an example of a *pattern of interplay* representing the payment of a meal at a restaurant. Three *pattern actions* are provided with *variations* and *exceptions*. The behavior associated to a *pattern action*, *variation,* or *exception* can be described by either an *event* or a more detailed *BOM*. *Event types* are further described in Section 6.2.4. While *events* and *BOMs* are used to represent *pattern actions, variations,* and *exceptions*, the actual behavior modeling required in carrying out a *pattern action, variation,* or *exception* by a federate is an implementation focus, which is outside the scope of this specification.

Table 6-7 Pattern of Interplay Table Example

Pattern of Interplay Name		Seq	Name	Sender	Receiver	Event	BOM	Condition	Note
Restaurant Payment	**Pattern Action**	1	Customer Done	CustomerEntity	Waiter Entity	na	na		*[2]
	Variation		Customer Idle	Table Entity	na	DirtyDishes OnTable	na	table has dirty dishes	na
	Variation		Customer Request	CustomerEntity	Waiter Entity	CheckRequest	na	na	na
	Pattern Action	2	Check Brought ToTable	Waiter Entity	CustomerEntity	PaymentDue Notification	na		na
	Pattern Action	3	Customer Pays	CustomerEntity	Waiter Entity	na	na		na
	Variation		BillPayed _Cash	CustomerEntity	Waiter Entity	na	CashPayment BOM	na	na
	Variation		BillPayed _Credit	CustomerEntity	Waiter Entity	na	CreditCard PaymentBOM	na	na
	Exception		BillNot Payed	CustomerEntity	na	NonPaying Customer	na	customer leaves without paying bill	na
	Pattern Action	4	Customer LeavesTip	CustomerEntity	Waiter Entity	Tip	na		na
Note	*[1]								

6.2.2 State Machine

6.2.2.1 Purpose/Background

The *state machine* template component provides a mechanism for identifying the behavior *states* expected to be exhibited by one or more *conceptual entities*.

6.2.2.2 Table Format

The State Machine Table is the template component of the BOM DIF used to identify *state machines* at the Conceptual Model Definition level and is provided in Table 6-8.

Table 6-8 State Machine Table Format

State Machine Name	Conceptual Entities	State		
		State Name	Exit Condition	
			Exit Action	Next State
<state machine name>	<entity type name>, <entity type name>	<state name>	<exit action name>	<next state name>
			<exit action name>	<next state name>
		<state name>	<exit action name>	<next state name>
<state machine name>	<entity type name>, <entity type name>	<state name>	<exit action name>	<next state name>
			<exit action name>	<next state name>
<state machine name>	<entity type name>	<state name>	<exit action name>	<next state name>
			<exit action name>	<next state name>

One or more *state machine(s)* can be identified by this table. As depicted in Table 6-9, each *state machine* is intended to identify the *states* that can be represented behaviorally by one or more *conceptual entities*. A *name* and one or more *exit condition(s)* are identified for each *state*. Each *exit condition* is represented by an *exit action,* which is associated with a *pattern action* found within *pattern of interplay* (see Section 6.2.1), and identifies the *next state* upon satisfying the *exit condition*.

Table 6-9 State Machine Information Description Table

Category	Description	Occurs	Values
State Machine	*Identifies a state machine.*	1..many	
Name	Identifies the name of the *state machine*.	1	*text*
Conceptual Entities	Identifies one or more conceptual *entity types* that can represent the *states* defined.	1..many	*entitytIdentifier*
State	Identifies a behavior *state* which a conceptual entity may exhibit.	1..many	
Name	Identifies the name of the *state* to be described.	1	*text*
Exit Condition	Identifies the condition for a state transition. Includes the identification of the *pattern action* that causes the change and which *state* is the result of the change.	0..many	
Exit Action	Identifies the *pattern action* found within a *pattern of interplay* in which the *exit condition* has been satisfied. Note: a *pattern action* is represented by a BOM *event* or a *BOM*.	1	*text*
Next State	Identifies which *state* succeeds the present *state* based on the condition of the *exit action*.	1	*text*

The relationship of the *state machine* sub-elements within a BOM is illustrated in Figure 6-5.

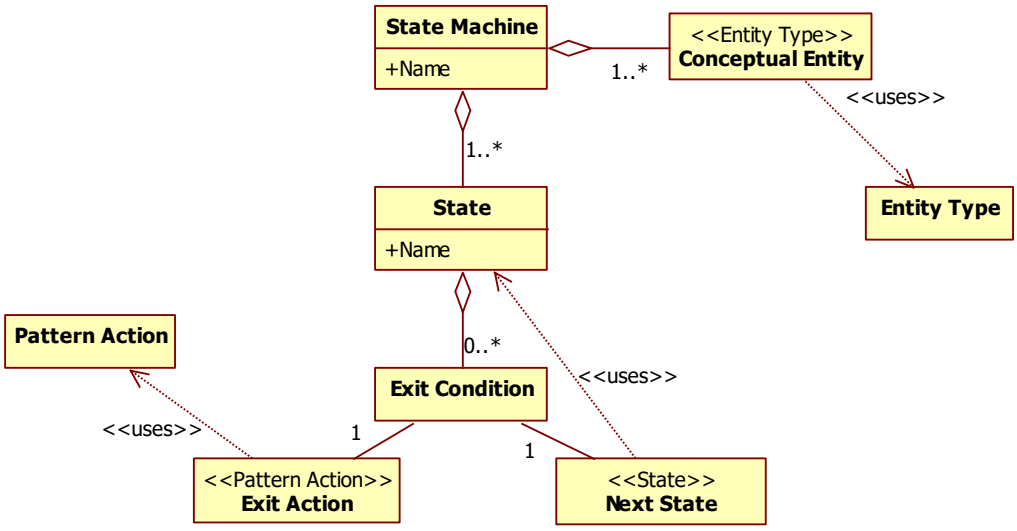


Figure 6-5 BOM State Machine Relationship

Figure 6-5 illustrates that a *state machine* consists of one or more *state(s)* and *conceptual entity(ies)*. A *conceptual entity* uses a specific BOM *entity type*. The *exit condition* for a *state* consists of an *exit action* and a *next state.* An *exit action* uses a specific *pattern action* defined within a *pattern of interplay*, whereas a *next state* uses another *state* within the *state machine*. Section 6.2.1 describes the template components needed

for defining BOM *pattern actions*. Section 6.2.3 describes the template components needed for defining BOM *entity types*.

A Dot Notation shall be used for *state machine* values as needed to ensure that the reference within the template component is unambiguous. Thus, *conceptual entity* and *exit action* names shall use Dot Notation as necessary, which may include the full parentage to uniquely identify the *entity type* or *pattern action* being used.

6.2.2.3 Inclusion Criteria

Every BOM that includes the Conceptual Model Definition section shall either contain a *state machine* or reference elements of a *state machine* captured within another BOM. The categories of information specified in Table 6-8 shall be included for all *state machine* elements unless "0..1" or "0..many" is identified in the Occurs column of Table 6-9.

6.2.2.4 Example

Table 6-10 gives an example of a *state machine* representing the employee of a restaurant. Six states are identified with reference to three *patterns of interplay*: Restaurant Visitation, Restaurant Payment, and Food Order. The interface details of the causes of a transition from a state is contained within each exit condition. Also included is the subsequent state for each of these exit conditions. The *pattern action* which causes an exit condition can be carried either via BOM event (see Section 6.2.4) or by a specific BOM. While the behavioral states of a conceptual entity are discussed in this specification, the actual behavior implementation necessary for modeling the states of a conceptual entity is outside the scope of a BOM.

Table 6-10 – State Machine Example

State Machine Name	Conceptual Entity	State			Note
		State Name	Exit Condition		
			Exit Action	Next State	
EmployeeSM	GreeterEntity, WaiterEntity	Ready	RestaurantVisitationBOM.CustomerArrives	Greet	*[5]
		Greet	RestaurantVisitationBOM.GreetCustomer	Seat	na
		Seat	RestaurantVisitationBOM.SeatCustomer	TakeOrder	na
		MonitorCustomer	RestaurantFoodPreparationBOM.CustomerOrders	ProcessOrder	na
			RestaurantPaymentBOM.CustomerRequest	PrepareBill	na
			RestaurantPaymentBOM.CustomerPays	ProcessBill	na
			RestaurantVisitationBOM.CustomerLeft	ClearingTable	na
		ProcessOrder	RestaurantFoodPreparationBOM.SubmitOrderToKitchen	MonitorKitchenOrder, MonitorCustomer	na
		MonitorKitchenOrder	RestaurantFoodPreparationBOM.FoodReadyToServe	Serve	na
		Serve	RestaurantFoodPreparationBOM.FoodBroughtToTable	MonitorCustomer	na
		PrepareBill	RestaurantPaymentBOM.CheckBroughtToTable	MonitorCustomer	
		ProcessBill	RestaurantPaymentBOM.ReceiptChangeReturned	MonitorCustomer	na
		ClearingTable	RestaurantPaymentBOM.TableCleared	Ready	na
Note	na				

6.2.3 Entity Type

6.2.3.1 Purpose/Background

The *entity type* template component provides a mechanism for describing the types of conceptual entities used to represent *senders* and *receivers* identified within a *pattern of interplay* and carry out the role of *conceptual entities* identified within a *state machine*.

6.2.3.2 Table Format

The Entity Type Definition Table is the template component of the BOM DIF used to identify *entity types* at the Conceptual Model Definition level and is provided in Table 6-11.

Table 6-11 Entity Type Table Format

Entity Type Name	Entity Type Characteristic Name
<entity type name>	<characteristic name>, <characteristic name>
<entity type name>	<characteristic name>, <characteristic name>

One or more *entity types* can be identified by the Entity Type Table. An *entity type* is intended to describe a *sender* or *receiver* associated with a *pattern action* of a *pattern of interplay* (see Section 6.2.1) and/or a *conceptual entity* associated with a *state machine* (see Section 6.2.2). As depicted in Table 6-12, each *entity type* is uniquely identified by a *name* and associated *characteristics*.

Table 6-12 Entity Type Information Description Table

Category	Description	Occurs	Values
Entity Type	Identifies an entity type within the conceptual model	1..many	
Name	Identifies a unique name for the BOM *entity type*	1	*text*
Characteristic	Identifies an attribute-like quality associated with the *entity type*	1..many	
Name	Identifies a unique name for the *characteristic*	1	*text*

The relationship of the *entity type* sub-elements within a BOM is illustrated in Figure 6-6.

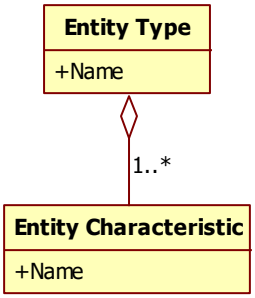


Figure 6-6 BOM Entity Type Relationship

Figure 6-6 illustrates that an *entity type* may consist of multiple *entity characteristics.* An *entity characteristic* may also be used by an *event type.* Section 6.2.4 describes the template components needed for identifying BOM *event types* (as either *triggers* or *messages*).

6.2.3.3 Inclusion Criteria

Every BOM that includes the Conceptual Model Definition section shall contain an *entity type* or reference an *entity type* defined within another BOM. The categories of information specified in Table 6-11 shall be included for all *entity types*, unless "0..1" or "0..many" is identified in the Occurs column of Table 6-12.

6.2.3.4 Example

Table 6-13 shows a simple example of the definition of *entity types* and their *characteristics*.

Table 6-13 Entity Type Definition Table Example

Entity Type Name	Entity Type Characteristic Name	Note
TableEntity	ID	na
	Occupied	na
	Dishes_Dirty	na
CustomerEntity	ID	na
	Table_ID	na
	FoodOrder	na
	PaidBill	na
WaiterEntity	ID	na
	Assigned_Tables	na
BillEntity	Amount	na
	Tip	na
	Waiter_ID	na
	Table_ID	na
Note	na	

6.2.4 Event Type Definition

6.2.4.1 Purpose/Background

The *event type* template component provides a mechanism for describing the types of *conceptual events* used to represent and carry out *pattern actions*, *variations*, and *exceptions* defined within a *pattern of interplay*. The

event type template component supports the ability to identify two categories of BOM *events*: *triggers* and *messages*.

Figure 6-7 illustrates that a BOM *pattern action*, *variation,* or *exception* can be addressed by either a *trigger* or a *message event type*. Furthermore, Figure 6-7 shows that the *event characteristics* of *triggers* and *messages* relate to the *sender* and *receiver* values that are identified for a *pattern action*, *variation,* or *exception.* Figure 6-7 also illustrates that the *event characteristics* (*source*, *target*, or *content*) for either a *message* or *trigger* are identified by *entity characteristics*.

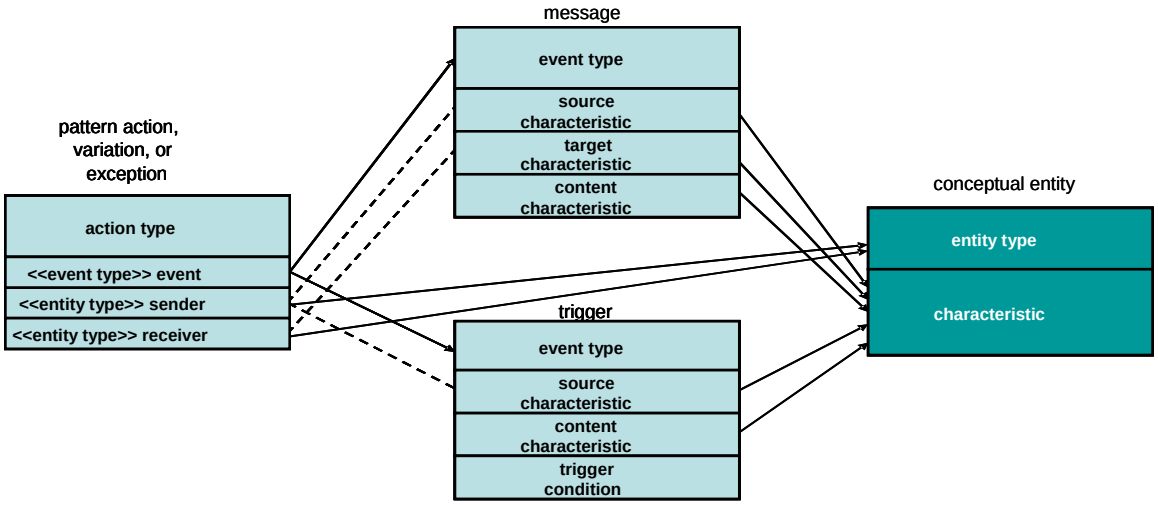


Figure 6-7 Pattern Action, Conceptual Event, and Conceptual Entity Association

The solid arrows in this figure indicate pointers to (a) an *event type*, such as a *message* or a *trigger*, which is associated to the *event* property of a *pattern action*, *variation,* or *exception*, (b) an *entity type* which is associated to the *sender and receiver* property of a *pattern action*, *variation,* or *exception,* and (c) an *entity's characteristics*, which are associated to the *characteristic* properties of an *event* such as a *message* or *trigger*. The dash lines in this figure indicate that the *characteristic* fields of an *event type* are indirectly associated with the *sender* and *receiver* parameters of a *pattern action*, *variation,* or *exception*.

The *sender* and *receiver* identified for a *pattern action*, *variation,* or *exception* is used to help understand the relationship of the *pattern of interplay* among the *conceptual entities* that are to be exhibited within a simulation or federation. However, the *event type* that may be used to carry out a *pattern action, variation,* or *exception* may or may not identify the intended *receiver(s)*. This depends upon whether the *event type* is characterized as a *message* or a *trigger.* Both *event types* shall have a *source characteristic*. *Triggers* do not have a *target characteristic* and may have a *content characteristic*. *Messages* have a *target characteristic*, do not have a *trigger characteristic*, and may have a *content characteristic*.

The *entity type* associated with an *event's source characteristic* should correspond with the *sender* value associated with the *pattern action*, *variation,* or *exception.* In the same way, the *entity type* associated with an *event's target characteristic*, which is found within a *message,* should correspond with the *receiver* value associated with the *pattern action*, *variation*, or *exception*.

Both *triggers* and *messages* are described in greater detail in the following sub-clauses.

6.2.4.1.1 BOM Triggers

When a *characteristic* value of a *conceptual entity* has changed, a result may be an undirected *event* in which a response (reaction) may occur by one or more other *conceptual entity(ies)* within the virtual environment that has interest in such observable changes. This type of *event* is known as a *trigger*; a term leveraged from the video game industry.

Within a virtual world such as depicted within a video game, there can be many types of game objects such as traps, exploding boxes, breakable crates, elevator switches, *or* guard dogs that react to the presence or location change of another game object such as a character. Such behavior is likely to occur in the simulation space used for DoD and/or commercial projects.

An *event type* that is a *trigger* is distinct because there is no intended *target entity* identified within the definition of the *event type*. This is because an *event type* that is a *trigger*, which is used to represent a *pattern action, variation, or exception*, identifies for its *characteristics* the *source entity*, and the *trigger condition*, but not the *target entity*.

Typically a *trigger* occurs within an HLA federation execution when either an HLA *object class attribute* has been updated or an HLA *interaction* has been sent, and the occurrence of that undirected *event* is of interest to a federate, which might react or respond to such a *pattern action*.

6.2.4.1.2 BOM Messages

A *message* is a BOM *event* directed between *conceptual entities*. An example would be a point-to-point phone call, which requires a phone to transmit data to another phone connected at a specified number. An *event type* that is a *message* is distinct because there is an intended *target entity* identified within the definition of the *event type*. This is because the *characteristics* of an *event type* that is a *message*, which is used to represent a *pattern action, variation, or exception*, uniquely identify instances of *source* and *target entities*.

Within an HLA federation execution, a *message* typically occurs between federates via an HLA Send Interaction or HLA Update Attributes Values invocation. Specifically it is an *event* intended for a known type of *conceptual entity*. The *conceptual entity* receiving the *message* is modeled in the simulation space by a federate in control of an HLA object instance. This federate will fulfill the *message* event by reacting or responding to a specific HLA Send Interaction or HLA Update Attributes Values invocation via *state* changes reflected in the corresponding HLA object instance.

6.2.4.2 Table Format

The Event Type Definition Table is the template component of the BOM DIF used to identify *event types* of a Conceptual Model Definition within a BOM and is provided in Table 6-14. The Event Type Definition Table may consist of one or more rows where each row represents an *event type*.

Table 6-14 Event Type Definition Table Format

Event Type Name	Source Characteristic Name	Target Characteristic Name	Content Characteristic Name	Trigger Condition
<event type name>	<source char>	<target char>, <target char>,	<content char>, <content char>	<trigger expression>, <trigger expression>,
<event type name>	<source char>	<target char>	<content char>, <content char>	<trigger expression>, <trigger expression>,
<event type name>	<source char>	<target char>, <target char>	<content char>, <content char>	<trigger expression>

One or more *event types* can be identified by the Event Type Table. An *event type* is used to characterize an *event* associated with a *pattern action* (see Section 6.2.2). As depicted in Table 6-15, the *event type* is uniquely identified by a *name, source*, *target* and *content characteristics,* and a *trigger condition*.

Table 6-15 Event Type Information Description Table

Category	Description	Occurs	Values
Event Type	Identifies an *event type* within the conceptual model.	0..many	
Name	Identifies a unique name for the BOM *event type*.	1	*text*
Source Characteristic	Identifies a *characteristic* defined within the Entity Type Table to be associated as the *source* of the *event type.*	1	
Name	References an *entity type characteristic* to be identified as the *source* of the *event.* The *entity type* associated to the *source characteristic* shall be identified using Dot Notation.	1	*text*
Target Characteristic	Identifies a *characteristic* defined within the Entity Type Table to be associated as a *target* of the *event type.*	0..many	
Name	References an *entity type characteristic* to be identified as the *target* of the *event.* The *entity type* associated to a *target characteristic* shall be identified using Dot Notation.	1	*text*
Content Characteristic	Identifies a *characteristic* defined within the Entity Type Table to be associated as a *content* of the *event type.*	0..many	
Name	References an *entity type characteristic* to be identified as the *content* of the *event.* The *entity type* associated to a *content characteristic* shall be identified using Dot Notation.	1	*text*
Trigger Condition	Specifies the condition in the form of a Boolean expression associated with triggering the *event type.*	0..1	*text*

The relationship of the *event type* sub-elements within a BOM is illustrated in Figure 6-8.

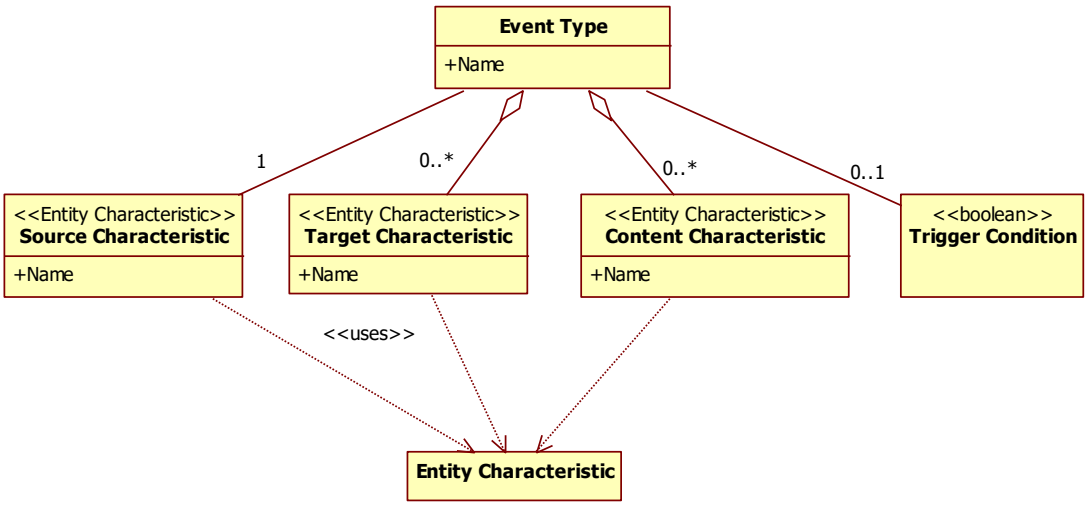


Figure 6-8 BOM Event Type Relationship

Figure 6-8 illustrates that an *event type* may consist of single *source characteristic,* multiple *target characteristics,* and *content characteristics,* and at least one *trigger condition*. A *source characteristic*, *target characteristic,* or *content characteristic* uses a specific BOM *entity characteristic*. Section 6.2.3 describes the template components needed for defining BOM *entity types* and their *entity characteristics*.

A Dot Notation shall be used for *event type* values as needed to ensure that the reference within the template component is unambiguous. Thus, *source characteristic*, *target characteristic,* or *content characteristic* names

shall use Dot Notation as necessary, which may include the full parentage to uniquely identify the *entity characteristic* being used.

6.2.4.3 Inclusion Criteria

Every BOM that includes the Conceptual Model Definition section shall contain an *event type* or reference an *event type* defined within another BOM. The categories of information specified in Table 6-14 shall be included for all *event types*, unless "0..1" or "0..many" is identified in the Occurs column of Table 6-15.

6.2.4.4 Example

Table 6-16 shows a simple example of how *event types* for both *triggers* and *messages* can be defined. If the BOM developer identifies a *target characteristic,* then the *event type* could be said to be a *message*. If there is no *target characteristic* but a *trigger condition* is expressed, then the *event type* is a *trigger*.

Table 6-16 Event Type Table Example

Event Type Name	Source Characteristic Name	Target Characteristic Name	Content Characteristic Name	Event Type Name	Note
DirtyDishesOnTable	TableEntity.ID	na	na	TableEntity.Dishes_Dirty = TRUE	na
NonPayingCustomer	CustomerEntity.ID	na	na	CustomerEntity.PaidBill = FALSE	*[3]
CheckRequest	CustomerEntity.ID	WaiterEntity.ID	na	na	*[4]
PaymentDueNotification	WaiterEntity.ID	CustomerEntity.ID	BillEntity.Amount	na	na
Tip	CustomerEntity.ID	WaiterEntity.ID	BillEntity.Tip	na	na
Note	na				

6.2.4.4.1 BOM Trigger Example

A change in the *characteristic* of a *conceptual entity* can be used to trigger *pattern action* behavior in other *conceptual entities*. For instance, in the **Restaurant Payment** *pattern of interplay* example provided in Table 6-7, the **CustomerIdle** *variation* is carried out by the **DirtyDishesOnTable** *event.* Examination of the **DirtyDishesOnTable** *event* in Table 6-16 reveals a *source characteristic* identified as **TableEntity.ID**, and a *trigger condition* of **Dishes_Dirty = true**. The *source characteristic* provides a pointer to an *entity type* responsible for the *event*; the **TableEntity** *entity type* is defined in Table 6-13, which also has a **Dishes_Dirty** *characteristic*. This *characteristic* corresponds with our ***event's trigger condition,*** **Dishes_Dirty = true**.

A *variation* such as **CustomerIdle** is used to identify one of the possible ways to complete a *pattern action*. The *pattern action* associated to this *variation* is identified as **CustomerDone**. The *sender* responsible for the **CustomerDone** *pattern action* is identified as **CustomerEntity**. And both the **CustomerDone** *pattern action* and **CustomerIdle** *variation* identified within the *pattern of interplay* reveals that **Waiter** is the anticipated *receiver.* Even though the **WaiterEntity** *entity type* was not identified for the **DirtyDishesOnTable** *event* as a *target characteristic*, the **DirtyDishesOnTable** *event* can still be used by the *pattern of interplay* as a way to *trigger* the **WaiterEntity** that the **CustomerEntity** is finished. What is characterized by this example is that the **TableEntity** *entity type* and its **Dishes_Dirty** *characteristic* have no knowledge of the **WaiterEntity,** however the **WaiterEntity** might certainly have knowledge of the **TableEntity** and may respond when **Dishes_Dirty = true**. Thus, for a *trigger*, it is only important to associate **TableEntity.ID** as the *source characteristic* and **TableEntity.Dishes_Dirty = true** as the *trigger condition* for the BOM *event*. In this *pattern*

of interplay example, the **WaiterEntity** then takes the next *pattern action* by notifying the **CustomerEntity** of the payment that is due through an *event* identified as **PaymentDueNotification**. **PaymentDueNotification**, which is an *event* that is a *message*, is further described in Section 6.2.4.4.2.

6.2.4.4.2 BOM Message Example

A *message* directed from one *conceptual entity* to another can be used as an *event* to carry out a *pattern action* behavior within a *pattern of interplay*. For instance, in the **Restaurant Payment** *pattern of interplay* example provided in Table 6-7, the **CheckBroughtToTable** *pattern action* is carried out by the **PaymentDueNotification** *event.* Examination of the **PaymentDueNotification** *event* in Table 6-16 reveals a *source characteristic* identified as **WaiterEntity.ID**, a *target characteristic* as **CustomerEntity.Table_ID,** and a *content characteristic* as **BillEntity.Amount**. Both the *source characteristic* and *target characteristic* provide a pointer to an *entity type* responsible for the *event*; the **WaiterEntity** and **CustomerEntity** *entity types* are defined in Table 6-13. The **WaiterEntity.ID** identifies the *source* responsible for the *event***.** The owner (parent) of this *source characteristic*, which is **WaiterEntity,** corresponds with the *sender* identified for the **CheckBroughtToTable** *pattern action*. Likewise, the **CustomerEntity.Table_ID** identifies the *target* for the *event*. Again, the owner (parent) of this *target characteristic*, which is **CustomerEntity**, corresponds with the *receiver* identified for the **CheckBroughtToTable** *pattern action*, and the **BillEntity.Amount** identifies the specific *message* content to be exchanged from the *source* to the *target*. This allows the **PaymentDueNotification** *message* to be directed from the **WaiterEntity** to **CustomerEntity** with the **BillEntity.Amount** exchanged. In this *pattern of interplay* example, the **CustomerEntity** then takes the next *pattern action* identified in Table 6-7, which is to pay the bill.

6.3 Model Mapping

The *model mapping* template component provides a mechanism for mapping the relationship between the entity and event elements of the Conceptual Model Definition (see Section 6.2), and the class structure elements of the Object Model Definition, which are described using HLA OMT Specification constructs (see Section 6.4). Two types of mappings are supported: *entity type mapping* and *event type mapping*.

6.3.1 Entity Type Mapping

6.3.1.1 Purpose/Background

The *entity type mapping* template component provides a mechanism for mapping between the *entity types* of a Conceptual Model Definition and the class structures of an Object Model Definition, which are described using HLA OMT Specification constructs.

6.3.1.2 Table Format

The Entity Type Mapping Table, provided in Table 6-17, is the template component of the BOM DIF used to map *entity types* and their associated *characteristics* to class structures defined by *HLA object classes* and their *attributes* or *HLA interactions classes* and their *parameters*.

Table 6-17 Entity Type Mapping Table Format

Entity Type Mapping		**Entity Characteristic Mapping**	
Entity Type Name	**HLA Object/Interaction Class**	**Entity Char Name**	**HLA Class Attribute/Parameter**
<entity type>	<HLA class>, <HLA class>	<char>	<HLA attrib/param>
		<char>	<HLA attrib/param>, <HLA attrib/param>
<entity type>	<HLA class>	<char>	<HLA attrib/param>
		<char>	<HLA attrib/param>

The Entity Type Mapping Table consists of a number of rows where each row represents a mapping of a Conceptual Model *entity type* to a corresponding Object Model Definition element. Object Model Definition elements are described using HLA OMT Specification constructs. These may include *HLA object classes* or *HLA interaction classes*, which are mapped with *entity types*, and the *HLA attributes* or *HLA parameters* associated with such an HLA OMT Specification class, which are mapped with a *characteristic* of an *entity type*.

As depicted in Table 6-18, the *entity type mapping* is intended to associate *entity types* and their *characteristics* with a representing *HLA object class* or *HLA interaction class* and their *HLA attributes* or *HLA parameters*.

Table 6-18 Entity Type Mapping Information Description Table

Category	**Description**	**Occurs**	**Values**
Entity Type Mapping	Identifies an *entity type mapping*.	1..many	

Name	References an *entity type* defined in the Entity Type Definition Table, which is to be mapped.	1	*text*
HLA Object/Interaction Class	References an *HLA object class* or *HLA interaction class* defined in either the current BOM or an external object model type (i.e., BOM, FOM, or SOM) that can be used to represent the specified *entity type*. If a class from an external object model type (i.e., BOM, FOM, or SOM) is being used, then a *note* reference shall be made for the element. This supporting *notes reference*, which is defined in the Notes Table, shall identify the *idtag* of the specific *HLA object class* or *HLA interaction class* defined in the external object model type (i.e., BOM, FOM, or SOM) URI convention as described in Section 6.5.1.	1..many	*text*
Characteristic	Identifies a *characteristic* associated with the *entity type* defined in the Entity Type Definition Table, which is to be mapped.	0..many	
Name	References a specific *characteristic* of the *entity type*.	1	*text*
HLA Attribute/Parameter	References an HLA object class *attribute* or HLA interaction class *parameter* that can be used to represent the specified entity type *characteristic*. An *attribute* or *parameter* can only be selected if its parent class, an *HLA object class* or *HLA interaction class*, has been identified. If multiple *HLA object classes* and/or *HLA interaction classes* have been identified that can represent the *entity type*, then the *HLA object class* or *HLA interaction class* associated with the *attribute* or *parameter* that can be used to represent the identified *characteristic* shall be identified using Dot Notation. If information from an external object model type (i.e., BOM, FOM, or SOM) is being used, then a *note* reference shall be made for the element. This supporting *notes reference*, which is defined in the Notes Table, shall identify the *idtag* of the specific *HLA attribute* or *HLA parameter* defined in the external object model type (i.e., BOM, FOM, or SOM) URI convention as described in Section 6.5.1.	1..many	*text*

The relationship of the *event type mapping* sub-elements within a BOM is illustrated in Figure 6-9.

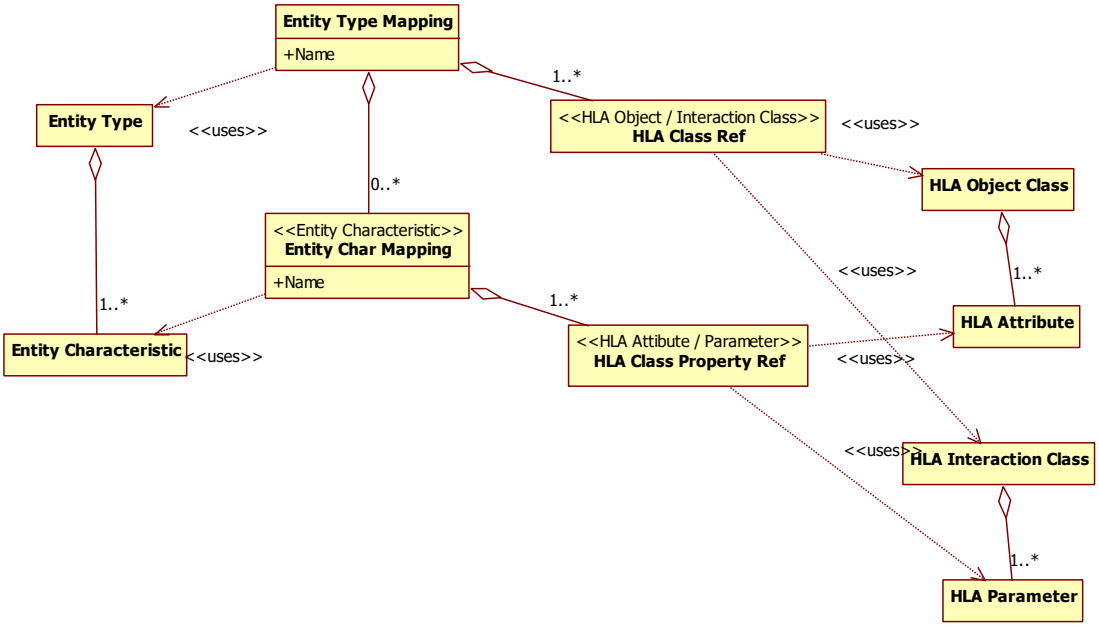


Figure 6-9 BOM Entity Type Mapping Relationship

Figure 6-9 illustrates that an *entity type mapping* may consist of a multiple *HLA class references* and *entity characteristic mapping.* An *entity characteristic mapping* may consist of multiple *HLA class property references*. An *entity type mapping* uses a specific BOM *entity type*. An *HLA class reference* uses a specific *HLA object class* or a specific *HLA interaction class*. An *entity characteristic mapping* uses an *entity*

characteristic defined within a BOM *entity type*. An *HLA class property reference* uses a specific *HLA attribute* or a specific *HLA parameter*. Section 6.2.3 describes the template components needed for defining BOM *entity types* and their *entity characteristics*. Section 6.4 and the HLA OMT Specification describe the template components needed for defining *HLA object classes*, *HLA interaction classes*, *HLA attributes,* and *HLA parameters*.

A Dot Notation shall be used for *entity type mapping* values as needed to ensure that the reference within the template component is unambiguous. Thus, *entity type mapping*, *entity char mapping*, *HLA class reference,* or *HLA class property reference* names shall use Dot Notation as necessary, which may include the full parentage to uniquely identify the *entity*, *entity characteristic, HLA object class, HLA interaction class, HLA attribute, or HLA parameter* being used.

6.3.1.3 Inclusion Criteria

Every BOM that includes a Conceptual Model Definition section containing an *entity type* may also include a Model Mapping section containing an *entity type mapping*. Likewise, every BOM that includes an Object Model Definition section containing a class structure element may also include a Model Mapping section containing an *entity type mapping*. It is also possible for a BOM to include only a Model Mapping section containing an *entity type mapping* in which the *entity types* and class structures being mapped are defined within other BOMs. The categories of information specified in Table 6-17 shall be included for all BOMs that contain an *entity type mapping,* unless "0..1" or "0..many" is identified in the Occurs column of Table 6-18.

6.3.1.4 Example

Table 6-19 shows a simple example of *entity type mappings*.

Table 6-19 Entity Type Mapping Example

Entity Type Mapping		Entity Characteristic Mapping		Note
Entity Type Name	**HLA Object/Interaction Class**	**Entity Characteristic Name**	**HLA Class Attribute/Parameter**	
TableEntity	HLAobjectRoot.PhysicalEntity.Table	ID	Table.ID	na
		Occupied	Table.isOccupied	na
		Dishes_Dirty	Table.hasDirtyDishes	na
CustomerEntity	HLAobjectRoot.PhysicalEntity.Human.Customer	ID	Customer.ID	
		Table_ID	Customer.AssignedTableID	na
		FoodOrder	Customer.Order	na
		PaidBill	Customer.hasPaid	na
WaiterEntity	HLAobjectRoot.PhysicalEntity.Human.Waiter	ID	Waiter.ID	na
		Assigned_Tables	Waiter.AssignedTables[]	na
BillEntity	HLAobjectRoot.PhysicalEntity.Payment	Amount	Payment.Amount	na
		Tip	Payment.Tip	na
	HLAobjectRoot.PhysicalEntity.Human.Waiter	Waiter_ID	Waiter.ID	na
	HLAobjectRoot.PhysicalEntity.Human.Customer	Table_ID	Customer.AssignedTableID	na
Name	na			

6.3.2 Event Type Mapping

6.3.2.1 Purpose/Background

The Event Type Mapping template component provides a mechanism for mapping between the *event types* of the Conceptual Model Definition and the class structures of an Object Model Definition, which are described using HLA OMT Specification constructs.

6.3.2.2 Table Format

The Event Type Mapping Table, provided in Table 6-20, is the template component of the BOM DIF used to identify *event type* mappings to class structures defined by *HLA object classes* and their *attributes* or *HLA interactions classes* and their *parameters*.

Table 6-20 Event Type Mapping Table Format

Event Type Mapping		Source Characteristic Mapping		Target Characteristic Mapping		Content Characteristic Mapping		Trigger Condition Mapping	
Name	**HLA Object / Interac-tion Classes**	**Name**	**HLA Attribs / Params**	**Name**	**HLA Attribs / Params**	**Name**	**HLA Attribs/ Params**	**Trigger Expression**	**HLA Attribs / Params**
<event type>	<HLA class>	<source char>	<HLA attrib/ param>	<target char>	<HLA attrib/ param>	<content char>	<HLA attrib/ param>	<trigger condition expression>	<HLA condition>
<event type>	<HLA class>	<source char>	<HLA attrib/ param>, <HLA attrib/ param>	<target char>	<HLA attrib/ param>	<content char>	<HLA attrib/ param>, <HLA attrib/ param>	<trigger condition expression>	<HLA condition>, <HLA condition>
				<target char>	<HLA attrib/ param>				
<event type>	<HLA class>	<source char>	<HLA attrib/ param>	<target char>	<HLA attrib/ param>	<content char>	<HLA attrib/ param>	<trigger condition expression>	<HLA condition>

The Event Type Mapping Table consists of a number of rows where each row represents a mapping of Conceptual Model Definition *event type* to a Object Model Definition element. Object Model Definition elements are described using HLA OMT Specification constructs. These may include *HLA object classes* or *HLA interaction classes*, which are mapped with *event types*, and the *HLA attributes* or *HLA parameters* associated with such an HLA OMT Specification class, which are mapped with a *source characteristic, target characteristic, content characteristic* or *trigger condition* of an *event type*.

As depicted in Table 6-21, the *event type mapping* is intended to associate *event types* and their *characteristics* with a representing *HLA object class* or *HLA interaction class* and their *HLA attributes* or *HLA parameters*.

Table 6-21 Event Type Mapping Information Description Table

Category	Description	Occurs	Values
Event Type Mapping	Identifies an *event type mapping*.	1..many	
Name	References an *event type* defined in the Event Type Definition Table.	1	*text*
HLA Object/Interaction Class	References an *HLA object class* or *HLA interaction class* defined in either the current BOM or in an external object model type (i.e., BOM, FOM, or SOM) that can be used to represent the specified *event type*. If a class from an external object model type (i.e., BOM, FOM, or SOM) is being used, then a *note* reference shall be made for the element. This supporting *notes reference*, which is defined in the Notes Table, shall identify the *idtag* of the specific HLA Object/Interaction Class defined in the external object model type (i.e., BOM, FOM, or SOM) URI convention as described in Section 6.5.1.	1..many	*text*
Source Characteristic	Identifies a *source characteristic* associated with the *event type* identified in the Event Type Definition Table to be mapped.	1	
Name	References a specific *characteristic* associated with the *event type*.	1	*text*
HLA Attribute/Parameter	References an HLA object class *attribute* or HLA interaction class *parameter* that can be used to represent the specified *event type source characteristic*. An *attribute* or *parameter* can only be selected if its parent class, an *HLA object class* or *HLA interaction class*, has been identified. If multiple *HLA object classes* and/or *HLA interaction classes* have been identified that can be used to represent the *event type*, then the *HLA object class* or *HLA interaction class* associated with the *attribute* or *parameter* that can be used to represent the identified *characteristic* shall be identified using Dot Notation. If information from an external object model type (i.e., BOM, FOM, or SOM) is being used, then a *note* reference shall be made for the element. This supporting *notes reference*, which is defined in the Notes Table, shall identify the *idtag* of the specific *HLA attribute* or *HLA parameter* defined in the external object model type (i.e., BOM, FOM, or SOM) URI convention as described in Section 6.5.1.	1..many	*text*
Target Characteristic	Identifies a *characteristic* defined within the Entity Type Table to be associated as a *target* of the *event type.*	0..many	
Name	References a specific *characteristic* associated with the *event type*.	1	*text*
HLA Attribute/Parameter	References an HLA object class *attribute* or HLA interaction class *parameter* that can be used to represent the specified *event type target characteristic*. An *attribute* or *parameter* can only be selected if its parent class, an *HLA object class* or *HLA interaction class*, has been identified. If multiple *HLA object classes* and/or *HLA interaction classes* have been identified that can represent the *event type*, then the *HLA object class* or *HLA interaction class* associated with the *attribute* or *parameter* that can be used to represent the identified *characteristic* shall be identified using Dot Notation. If information from an external object model type (i.e., BOM, FOM, or SOM) is being used, then a *note* reference shall be made for the element. This supporting *notes reference*, which is defined in the Notes Table, shall identify the *idtag* of the specific *HLA attribute* or *HLA parameter* defined in the external object model type (i.e., BOM, FOM, or SOM) URI convention as described in Section 6.5.1.	1..many	*text*
Content Characteristic	Identifies a *characteristic* defined within the Entity Type Table to be associated as a *content* of the *event type.*	0..many	
Name	References a specific *characteristic* associated with the *event type*.	1	*text*
HLA Attribute/Parameter	References an HLA object class *attribute* or HLA interaction class *parameter* that can be used to represent the specified *event type content characteristic*. An *attribute* or *parameter* can only be selected if its parent class, an *HLA object class* or *HLA interaction class*, has been identified. If multiple *HLA object classes* and/or *HLA interaction classes* have been identified that can represent the *event type*, then the *HLA object class* or *HLA interaction class* associated with the *attribute* or *parameter* that can be used to represent the identified *characteristic* shall be identified using Dot Notation. If information from an external object model type (i.e., BOM, FOM, or SOM) is being used, then a *note* reference shall be made for the element. This supporting *notes reference*, which is defined in the Notes Table, shall identify the *idtag* of the specific *HLA attribute* or *HLA parameter* defined in the external object model type (i.e., BOM, FOM, or SOM) URI convention as described in Section 6.5.1.	1..many	*text*
Trigger Condition	Specifies the condition in the form of a Boolean expression associated with triggering the *event type.*	0..1	
Name	References a specific *trigger condition expression* associated with the *event type*.	1	*text*

HLA Attribute/Parameter	References HLA object class *attribute(s)* or HLA interaction class *parameter(s) that are used to state the trigger condition.* An *attribute* or *parameter* can only be selected if its parent class, an *HLA object class* or *HLA interaction class*, has been identified. If multiple *HLA object classes* and/or *HLA interaction classes* have been identified that can represent the *event type*, then the *HLA object class* or *HLA interaction class* associated with the *attribute* or *parameter* that can be used to represent the identified *characteristic* shall be identified using Dot Notation. If information from an external object model type (i.e., BOM, FOM, or SOM) is being used, then a *note* reference shall be made for the element. This supporting *notes reference*, which is defined in the Notes Table, shall identify the *idtag* of the specific *HLA attribute* or *HLA parameter* defined in the external object model type (i.e., BOM, FOM, or SOM) URI convention as described in Section 6.5.1.	1..many	*text*

The relationship of the *event type mapping* sub-elements within a BOM is illustrated in Figure 6-10.

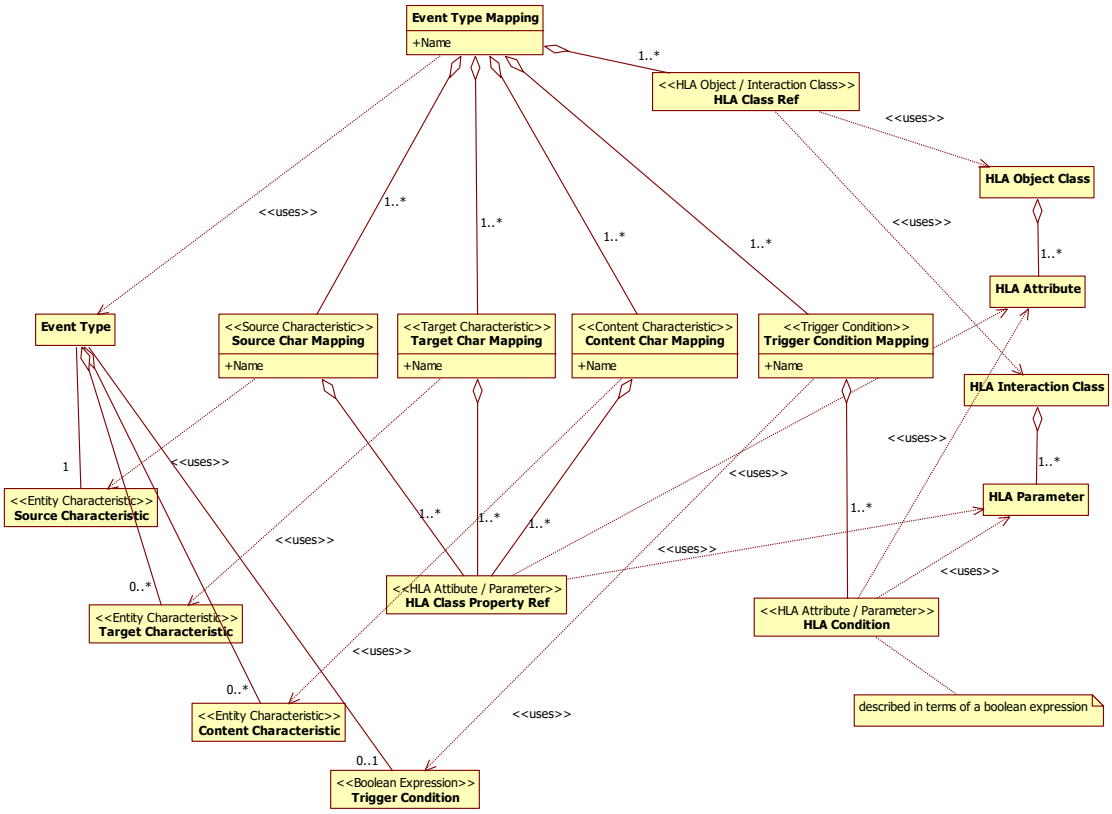


Figure 6-10 BOM Event Type Mapping Relationship

Figure 6-10 illustrates that an *event type mapping* may consist of a multiple *HLA class references*, *source characteristic mappings, target characteristic mappings, content characteristic mappings,* and *trigger condition mappings.* A *source characteristic mapping, target characteristic mapping,* or *content characteristic mapping* may consist of multiple *HLA class property references*. A *trigger condition mapping* may consist of multiple *HLA conditions.* An *event type mapping* uses a specific BOM *event type*. An *HLA class reference* uses a specific *HLA object class* or a specific *HLA interaction class*. A *source characteristic mapping, target characteristic mapping,* or *content characteristic mapping* uses a *source characteristic, target source, or content characteristic* identified within a BOM *event type*. Likewise, a *trigger condition mapping* uses a *trigger condition* identified within a BOM *event type*. An *HLA class property reference* uses a specific *HLA attribute* or a specific *HLA parameter*. Likewise, an *HLA condition* uses a specific *HLA attribute* or a specific *HLA parameter* in the form of a Boolean expression. Section 6.2.4 describes the template components needed for

defining BOM *event types* and their *characteristics*. Section 6.4 and the HLA OMT Specification describe the template components needed for defining *HLA object classes*, *HLA interaction classes*, *HLA attributes,* and *HLA parameters*.

A Dot Notation shall be used for *event type mapping* values as needed to ensure that the reference within the template component is unambiguous. Thus, *event type mapping*, *source characteristic mapping, target characteristic mapping, content characteristic mapping*, *trigger condition mapping, HLA class reference,* or *HLA class property reference* names shall use Dot Notation as necessary, which may include the full parentage to uniquely identify the *event, event characteristic, HLA object class, HLA interaction class, HLA attribute, or HLA parameter* being used.

6.3.2.3 Inclusion Criteria

Every BOM that includes a Conceptual Model Definition section containing an *event type* may also include a Model Mapping section containing an *event type mapping*. Likewise, every BOM that includes an Object Model Definition section containing a class structure element may also include a Model Mapping section containing an *event type mapping*. It is also possible for a BOM to include only a Model Mapping section containing an *event type mapping* in which the *event types* and class structures being mapped are defined within other BOMs. The categories of information specified in Table 6-20 shall be included for all BOMs that contain an *event type mapping*, unless “0..1” or “0..many” is identified in the Occurs column of Table 6-21.

6.3.2.4 Example

Table 6-22 shows a simple example of *event type mappings*.

Table 6-22 Event Type Mapping Example

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 Cond-ition**	na
DirtyDishesOnTable	HLAobjectRoot. PhysicalEntity.Table	TableEntity.ID	Table.ID	na	na	na	na	TableEntity. Dishes_Dirty = TRUE	Table. hasDirtyDishes == TRUE	na
NonPayingCustomer	HLAinteractionRoot. PhysicalEvent. CustomerLeaves	CustomeEntity.ID	CustomerLeaves. CustomerID	na	na	na	na	CustomerEntity. PaidBill = FALSE	Customer. hasPaid == FALSE	na
	HLAobjectRoot. PhysicalEntity. Customer		Customer.ID							
CheckRequest	RealWorldMessage. RequestBill	CustomerEntity.ID	RequestBill. CustomerID	WaiterEntity.ID	RequestBill. WaiterID	na	na	na	na	na
PaymentDueNotification	RealWorldMessage. PaymentDue	WaiterEntity.ID	PaymentDue. WaiterID	CustomerEntity. ID	PaymentDue. CustomerID	BillEntity.Amount	PaymentDue. Amount	na	na	na
Tip	HLAobjectRoot. PhysicalEntity. Payment	CustomerEntity. ID	Payment.Source	WaiterEntity.ID	Payment.Recipient	BillEntity.Tip	Payment.Tip	na	na	na
Note	na									

6.4 Object Model Definition

6.4.1.1 Purpose/Background

The Object Model Definition defines the structure of object and interaction classes, and their associated attributes and parameters, which could be used in defining the interface to a simulation or federation that implements the conceptual model. The specific mapping of class structure elements described in an Object Model Definition to the *entity* and *event* elements described in a Conceptual Model Definition (see Section 6.2) is supported using the Model Mapping (see Section 6.3).

6.4.1.2 Table Format

The Object Model Definition elements are described using *HLA object classes* and their corresponding *HLA attributes*, *HLA interaction classes* and their corresponding *HLA parameters*, and *HLA datatypes*. The template components for each of these Object Model Definition elements are fully defined in the HLA OMT Specification.

The relationship of the Object Model Definition elements within a BOM is illustrated in Figure 6-11.

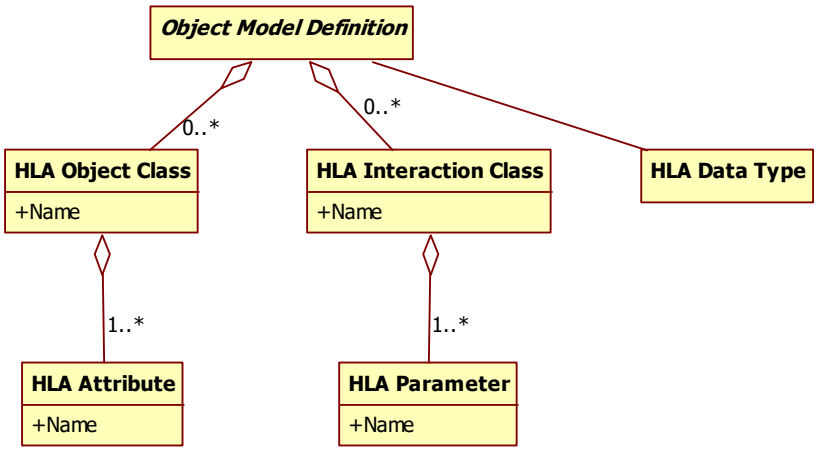


Figure 6-11 BOM Object Model Definition Relationship

Figure 6-11 illustrates that the Object Model Definition elements may consist of multiple *HLA object classes*, *HLA interaction classes*, or *HLA data types*. An *HLA object class* may consist of multiple *HLA attributes*. Likewise, an *HLA interaction class* may consist of multiple *HLA parameters.*

6.4.1.3 Inclusion Criteria

Every BOM that includes the Object Model Definition section shall contain at least one *HLA object class* or *HLA interaction class* with supporting *HLA attributes* or *HLA parameters*.

6.4.1.4 Example

Tables 6-23 through 6-26 provide examples of the Object Model Definition elements used to represent the Restaurant Payment BOM. Specific rules for filling in the Object Model Definition tables can be found in Section 8.4 of the "Guide for BOM Use and Implementation."

Table 6-23 HLA Object Classes Example

Object Class Definition			Note
HLAObjectRoot (N)	PhysicalEntity	Customer (PS)	na
		Waiter (PS)	na
		Table (PS)	na
	AbstractEntity	Payment (PS)	na
Note	na		

Table 6-24 HLA Attributes Example

Object Class Ref	Attribute	Datatype	Update Type	Update Condition	D/A	P/S	Available Dimensions	Transportation	Order	Note
HLAobjectRoot	HLAprivilege ToDeleteObject	na	na	na	na	na	na	na	na	na
PhysicalEntity	ID	ID	Static	na	na	na	na	na	na	na
PhysicalEntity. Table	hasDirtyDishes	Boolean	Condi-tional	on change or request	na	na	na	na	na	na
	isOccupied	Boolean	Condi-tional	on change or request	na	na	na	na	na	na
PhysicalEntity. Waiter	AssignedTables	Assigned TablesType	Condi-tional	on change or request	na	na	na	na	na	na
PhysicalEntity. Customer	TableID	ID	Static	na	na	na	na	na	na	na
	Order	OrderType	Condi-tional	on change or request	na	na	na	na	na	na
	hasPaid	Boolean	Condi-tional	on change or request	na	na	na	na	na	na
AbstractEntity. Payment	Amount	Float32BE	Condi-tional	on change or request	na	na	na	na	na	na
	Tip	Float32BE	Condi-tional	on change or request	na	na	na	na	na	na
	Source	ID	Static	na	na	na	na	na	na	na
	Recipient	ID	Static	na	na	na	na	na	na	na
Note	na									

In this example, **HLAobjectRoot.PhysicalEntity.Waiter** object class defined in Table 6-23 inherits the **PhysicalEntity** attribute **ID** identified in Table 6-24 since **PhysicalEntity** is its parent. The **Table** object class also inherits this attribute, but extends it with two of its own with the **hasDirtyDishes** and **isOccupied** attributes. Additionally, the **Customer** object class also inherits the **ID** attribute, but extends it with one of its own with the **hasPaid** attribute.

Table 6-25 HLA Interaction Classes Example

Interaction Class Definition			Note
HLAinteractionRoot (N)	Physical Event	CustomerLeaves	na
	RealWorldMessage	RequestBill	na
		PaymentDue	na
Note	na		

Table 6-26 HLA Parameters Example

Interaction Class Ref	Parameter	Datatype	Available Dimensions	Transportation	Order	Note
HLAinteractionRoot. PhysicalEvent	CustomerID	ID	na	na	na	na
HLAinteractionRoot. RealWorldMessage	CustomerID	ID	na	na	na	na
RealWorldMessage. PaymentDue	WaiterID	ID	na	na	na	na
	Amount	Float32BE				
RealWorldMessage. RequestBill	WaiterID	ID	na	na	na	na
Note	na					

In this example, **HLAinteractionRoot.PhysicalEvent.CustomerLeaves** interaction class defined in Table 6-25 inherits the **PhysicalEvent** parameter **CustomerID** identified in Table 6-26 since **PhysicalEvent** is its parent. The **RequestBill** interaction class also inherits a **CustomerID** parameter from the **RealWorldMessage**, but extends it with two of its own with the **WaiterID** and **Amount** parameters.

6.5 Supporting Tables

Elements of a BOM can be clarified with the use of *notes* and *lexicon definitions*, which are a part of the Supporting Tables. These template components that may be used within a BOM are illustrated in Figure 6-12.

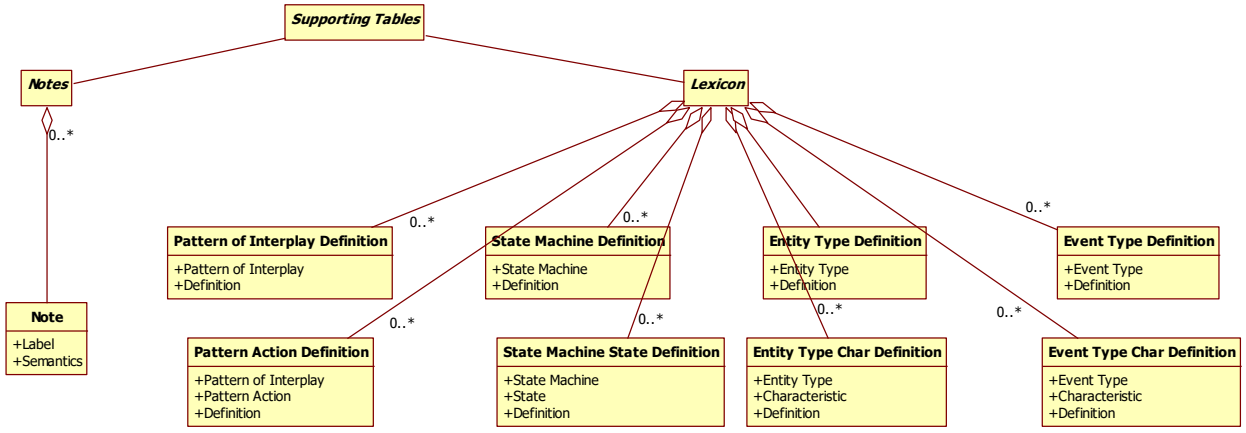


Figure 6-12 BOM Supporting Tables Elements Relationship

Figure 6-12 illustrates that the Supporting Tables may consist of *notes* and *lexicons*. The *lexicon* may consist of multiple definition entries for any of the BOM tables.

6.5.1 BOM Notes

6.5.1.1 Purpose/Background

A BOM table may be annotated with additional descriptive information outside of the immediate table structure by way of a *notes reference*. Notes can be attached to any cell of any table by way of a *notes reference*. As can be seen in the examples for Tables 6-7, 6-8, 6-9, and 6-11, the BOM Pattern of Interplay Table, State Machine Table, Event Table, and Entity Table have a *notes* column, which allows the association of a *note* (or *notes*) with an entire row of information, and a *notes* row, which allows the association of a *note* (or *notes*) with an entire table. This *notes* feature permits users to associate explanatory information with individual tables and sub-tables to facilitate effective use of the data. Additionally, following the convention of the HLA OMT Specification, the mechanism for attaching one or more *notes* to a BOM table entry shall be to include a notes pointer in the appropriate table cell. In the tabular BOM format, this *notes* pointer shall consist of a uniquely identifying note label (or a series of comma-separated labels) preceded by an asterisk and enclosed by brackets. The *notes* themselves shall be associated with the note label and included in the Notes Table. A single *note* may be referenced multiple times in a BOM.

6.5.1.2 Table Format

The table format for the Notes Table is provided in the HLA OMT Specification (Section 4.13.2 – Table 35).

A *note* value associated to the *notes reference* is defined within a free text field, however if a Uniform Resource Identifier (URI) is intended to be used to identify an external reference, then the following convention shall be used for the *note* value:

```
<scheme>://<authority><path>?<query>
```

Table 6-27 provides a description of each of these URI parts that can be used for a *note* value.

Table 6-27 URI Parts

URI Part	Description
<scheme>	Denotes the transport protocols like http, ftp, news, mailto, or file. This URI part is mandatory.
<authority>	Identifies a unique name assigned by a registration authority for resources. This URI part is optional.
<path>	Provides the resource identification within the realms of the <scheme> and the <authority>. This URI part is commonly used to identify a folder location or web site.
?<query>	Provides an opportunity to link to a specific portion of an XML (or HTML) document.

Two examples of the URI format in the context of a BOM note are provided as follows:

```
"file:c:\boms\bom12.xml#referencedObjectIdtag"

"http://www.bomscentral.com/bom34.xml#refClass"
```

Both `#referencedObjectIdtag` and `#refClass` are examples of the `?<query?>` value, and are commonly referred to as anchors or bookmarks. Anchors or bookmarks are commonly used in HTML. An *idtag* defined within a BOM, FOM, or SOM allows such anchors or bookmarks to be identified. An example of an *idtag* in a BOM is provided as follows:

```
<objectClass idtag = "referencedObjectIdtag">
            <name>MyClass</name>
```

The benefit in this convention is that it offers a capability for tools to provide automation in accessing external references.

6.5.1.3 Inclusion Criteria

BOMs describing *patterns of interplay*, *state machines, entity types, event types,* model mappings, and/or class structures may include *notes* wherever such annotation improves the clarity and understandability of the Conceptual Model Definition, Model Mapping, or Object Model Definition.

6.5.1.4 Example

Table 6-28 provides an example of the use of the notes feature. Tables 6-7, 6-10, and 6-16 provide examples containing references to the notes in this table.

Table 6-28 Notes Table Example

Notes	
Label	**Semantics**
1	This pattern of interplay might be able to be used as a template to support payment patterns for other service oriented businesses
2	This pattern action may involve other employees including the host (i.e., Greeter)
3	The support for this trigger may involve a completely other BOM (pattern) to support non-paying customer
4	Customer may also request check from other employee conceptual entity types
5	This is the idle state for the Greeter or Waiter

6.5.2 BOM Lexicon Definitions

6.5.2.1 Purpose/Background

If the reuse of conceptual models and components is to be achieved among simulations, it is necessary not only to specify the classes of data required by the BOM template components identified in Section 6, but also to achieve a common understanding of the semantics of this data. By maintaining separate Lexicon Tables, the BOM *lexicon definitions* provides a means for simulation software designers to define all *entity types*, *event types*, *entity type characteristics*, *event type characteristics*, *pattern actions*, and *state machine states* in BOMs. For class structures such as *HLA object classes*, *HLA interaction classes*, HLA object class *attributes*, and HLA interaction class *parameters*, the Lexicon Tables provided in the HLA OMT Specification shall be used.

6.5.2.2 Pattern of Interplay Definition Table

This sub-paragraph provides the format for describing the semantics for *patterns of interplay*. The template that shall be used for this information is provided in Table 6-29.

Table 6-29 BOM Pattern of Interplay Description Definition Table Format

Pattern of Interplay	Definition
<pattern of interplay>	<definition>
...	...
<pattern of interplay>	<definition>

- The first column (Pattern of Interplay) shall contain the names of the *patterns of interplay* described in the BOM.
- The second column (Definition) shall describe the semantics for the *pattern of interplay*.

Table 6-30 provides an example of the use of the table. The *pattern of interplay* is taken from the examples provided earlier in Section 6.2.1.

Table 6-30 BOM Pattern of Interplay Definition Table Example

Pattern of Interplay	Definition
RestaurantPayment	Payment of a meal at a restaurant
RestaurantVisitation	Customer visitation activities at a restaurant
FoodOrder	Actions associated with completing a food order
AbstractEntity.CashPayment	Represents a payment using cash

6.5.2.3 Pattern Action Definition Table

This sub-paragraph provides the format for describing the semantics for *pattern actions*, which include *exception* and *variation* elements. The template that shall be used for this information is provided in Table 6-31.

Table 6-31 BOM Pattern Action Definition Table Format

Pattern of Interplay	Pattern Action	Definition
<pattern of interplay>	<action>	<definition>
...	...	...
<pattern of interplay>	<action>	<definition>

- The first column (Pattern of Interplay) shall contain the names of the *patterns of interplay* described in the BOM.
- The second column (Pattern Action) shall contain the name of the *pattern action* associated with a *pattern of interplay*, which may include an *exception* or *variation*. For *exceptions* and *variations*, Dot Notation shall be used to identify the associated *pattern action*.
- The third column (Definition) shall describe the semantics for the *pattern action*.

Table 6-32 provides an example of the use of the table. The *pattern actions* are taken from the examples provided earlier in Section 6.2.1.

Table 6-32 BOM Pattern Action Definition Table Example

Pattern of Interplay	Action	Definition
RestaurantPayment	CustomerDone	Waiter is made aware that Customer will not order anything else
RestaurantPayment	CustomerDone.CustomerIdle	Waiter observes that the table is full of dirty dishes
RestaurantPayment	CustomerDone.CustomerRequest	Waiter is called by Customer
RestaurantPayment	CheckBroughtToTable	Waiter notifies Customer of payment due
RestaurantPayment	CustomerPays	Customer pays bill

RestaurantPayment	CustomerPays.BillPayed_Cash	Customer pays by cash
RestaurantPayment	CustomerPays.BillPayed_Credit	Customer pays by credit card
RestaurantPayment	CustomerPays.BillNotPayed	Customer leaves without paying

6.5.2.4 State Machine Description Definition Table

This sub-paragraph provides the format for describing the semantics for *state machines*. The template that shall be used for this information is provided in Table 6-33.

Table 6-33 BOM State Machine Definition Table Format

State Machine	Definition
<state machine>	<definition>
...	...
<state machine>	<definition>

- The first column (State Machine) shall contain the names of the *state machines* described in the BOM.
- The second column (Definition) shall describe the semantics for the *state machine*.

Table 6-34 provides an example of the use of the table. The *state machines* are taken from the examples provided earlier in Section 6.2.2.

Table 6-34 BOM State Machine Description Definition Table Example

State Machine	Definition
EmployeeSM	Employee of a restaurant
CustomerSM	Common customer actions

6.5.2.5 State Machine State Definition Table

This sub-paragraph provides the format for describing the semantics for *state machine states*. The template that shall be used for this information is provided in Table 6-35.

Table 6-35 BOM State Machine States Definition Table Format

State Machine	State	Definition
<state machine>	<state>	<definition>
...	...	...
<state machine>	<state>	<definition>

- The first column (State Machine) shall contain the names of the *state machines* described in the BOM.
- The second column (State) shall contain the name of the *state* associated with the *state machine*.
- The third column (Definition) shall describe the semantics for the *state*.

Table 6-36 provides an example of the use of the table. The *states* are taken from the examples provided earlier in Section 6.2.2.

Table 6-36 BOM State Machine States Definition Table Example

State Machine	State	Definition

EmployeeSM	Ready	Employee is ready to serve
EmployeeSM	Greet	Employee greets customer
EmployeeSM	ProcessBill	Employee processes bill
EmployeeSM	PrepareBill	Employee prepares bill
EmployeeSM	ProcessOrder	Employee processes food order
EmployeeSM	ClearingTable	Employee clears table

6.5.2.6 Entity Type Definition Table

This sub-paragraph provides the format for describing the semantics for *entity types*. The template that shall be used for this information is provided in Table 6-37.

Table 6-37 BOM Entity Type Definition Table Format

Entity Type	Definition
<entity type>	<definition>
...	...
<entity type>	<definition>

- The first column (Entity Type) shall contain the names of the *entity types* described in the BOM.
- The second column (Definition) shall describe the semantics for the *entity type*.

Table 6-38 provides an example of the use of the table. The *entity types* are taken from the examples provided earlier in Section 6.2.3.

Table 6-38 BOM Entity Type Definition Table Example

Type	Definition
TableEntity	Place where meal is served
CustomerEntity	Person paying for a meal
WaiterEntity	Waiter who served the meal and processes payment
BillEntity	Money owed for meal

6.5.2.7 Entity Type Characteristic Definition Table

This sub-paragraph provides the format for describing the semantics for *entity type characteristics*. The template that shall be used for this information is provided in Table 6-39.

Table 6-39 BOM Entity Type Characteristic Definition Table Format

Entity Type	Characteristic	Definition
<entity type>	<char>	<definition>
...	...	...

<entity type>	<char>	<definition>

- The first column (Entity Type) shall contain the names of the *entity types* described in the BOM.
- The second column (Characteristic) shall contain the name of the *characteristic* associated with the *entity type*.
- The third column (Definition) shall describe the semantics for the *characteristic*.

Table 6-40 provides an example of the use of the table. The *entity type characteristics* are taken from the examples provided earlier in Section 6.2.3

Table 6-40 BOM Entity Type Characteristic Definition Table Example

Type	Characteristic	Definition
TableEntity	Dishes_Dirty	That fact that the table has dirty dishes
CustomerEntity	ID	A Customer identifier
	TableID	The table Customer is at
WaiterEntity	ID	A Waiter identifier
	Assigned_Tables	List of tables for which Waiter is responsible
BillEntity	Amount	Amount of money
	Waiter_ID	Identifies Waiter processing payment
	Table_ID	Identifies Table at which the customer has been seated

6.5.2.8 Event Type Definition Table

This sub-clause provides the format for describing the semantics for *event types*. The template that shall be used for this information is provided in Table 6-41.

Table 6-41 – BOM Event Type Definition Table Format

Type	Definition
<event type>	<definition>
...	...
<event type>	<definition>

- The first column (Event Type) shall contain the names of all *event types* described in the BOM.
- The second column (Definition) shall describe the semantics for the *event type*.

Table 6-42 provides an example of the use of the table. The *event types* are taken from the examples provided earlier in Section 6.2.4

Table 6-42 BOM Event Type Definition Table Example

Type	Definition
DirtyDishesOnTable	There are dirty dishes on a table
NonPayingCustomer	Customer leaves without paying
CheckRequest	Request check from waiter
PaymentDueNotification	Notification of payment due

6.5.2.9 Event Type Characteristic Definition Table

This sub-paragraph provides the format for describing the semantics for *event type characteristics* including *source characteristics*, *target characteristics*, *content characteristics*, and *trigger conditions*. The template that shall be used for this information is provided in Table 6-43.

Table 6-43 BOM Event Type Characteristic Definition Table Format

Type	Characteristic	Definition
<event type>	<char>	<definition>
...	...	...
<event type>	<char>	<definition>

- The first column (Event Type) shall contain the names of all *event types* described in the BOM.
- The second column (Characteristic) shall contain the name of the *characteristic* associated with the *event type*.
- The third column (Definition) shall describe the semantics for the *characteristic*.

Table 6-44 provides an example of the use of the table. The *event type characterictics* are taken from the examples provided earlier in Section 6.2.4.

Table 6-44 BOM Event Type Characteristic Definition Table Example

Type	Characteristic	Definition
DirtyDishesOnTable	TableEntity.ID	Identifies Table
	TableEntity.Dishes_Dirty	Table has dirty dishes
NonPayingCustomer	CustomerEntity.ID	Identifies Customer
	CustomerEntity.hasPaid	Customer did not pay
CheckRequest	CustomerEntity.ID	Identifies Customer making the request
	WaiterEntity.ID	Identifies Waiter to whom the request is directed
PaymentDueNotification	WaiterEntity.ID	Identifies Waiter handling payment
	CustomerEntity.ID	Identifies Customer owing payment
	Amount	The total amount

7 BOM DIF Schema

The BOM template is encoded using XML. A BOM XML Schema has been defined to support the creation and validation of BOMs. Figure 7-1 provides a top-level graphical illustration of the BOM DIF schema hierarchy. See Annex C for an explanation of the graphical notations used in this section.

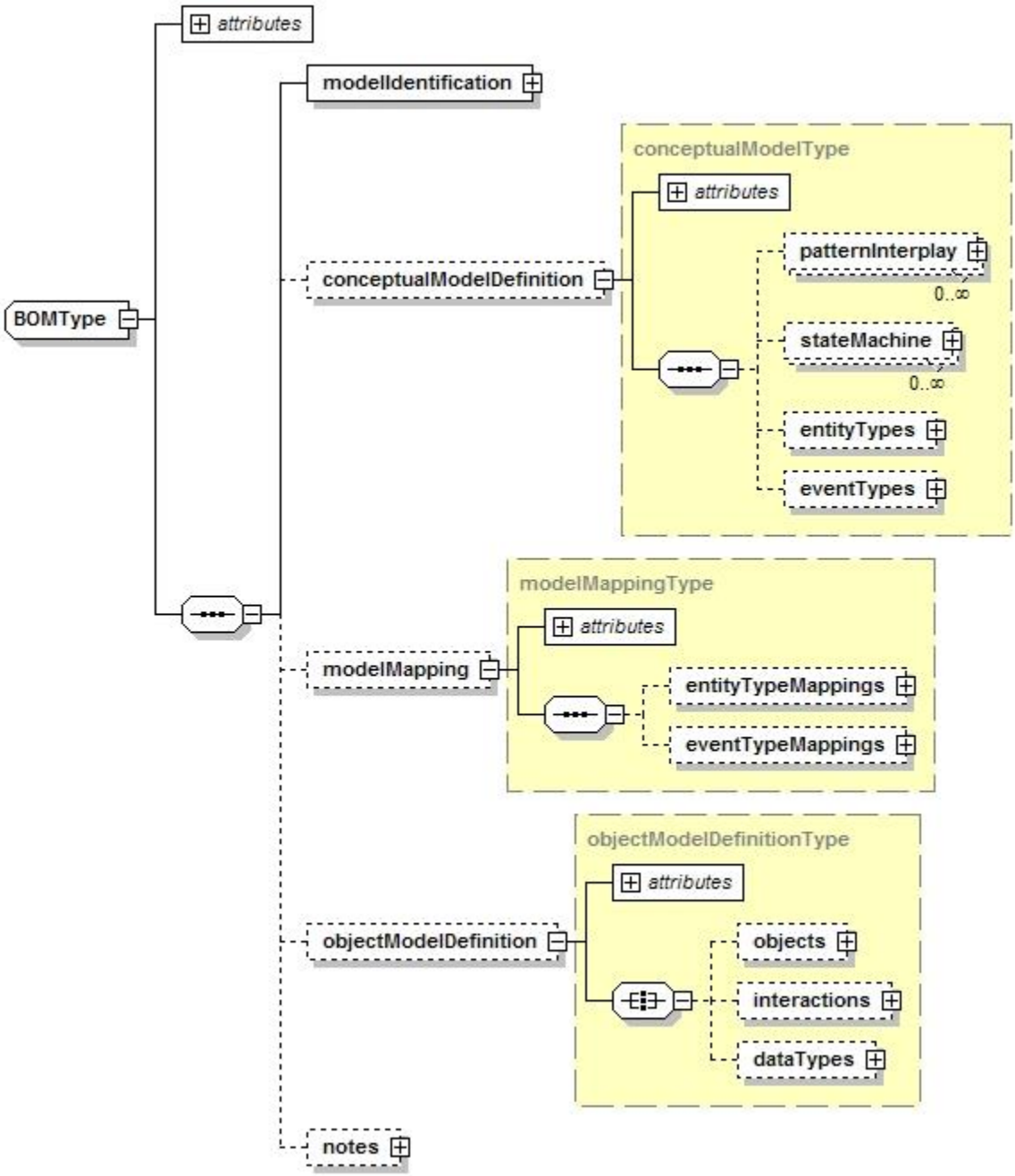


Figure 7-1 BOM DIF Schema Hierarchy

The major elements of the BOM schema are identified as:

- *modelIdentification*
  - *poc*
  - *keyword*
  - *reference*
  - *glyph*
- *conceptualModelDefinition*
  - *patternInterplay*
  - *stateMachine*
  - *entityType*
  - *eventType*
- *modelMapping*
  - *entityTypeMapping*
  - *eventTypeMapping*
- *objectModelDefinition*
  - *objectClasses*
  - *objectClassAttributes*
  - *interactionClasses*
  - *interactionClassParameters*
  - *data types*
- *notes*

The “attributes” node, which is identified in several instances within Figure 7-1 and is expanded in Figure 7-2, are the common XML attributes used to represent each of these major elements. These “attributes” include a *notes* reference, an *idtag* for supporting cross referencing by other elements and BOMs, and an *##other* attribute, which is used to capture additional information that may be associated with the BOM template component element.

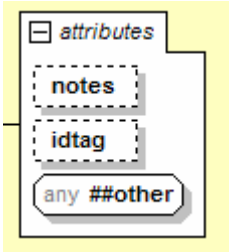


Figure 7-2 BOM DIF Schema Common Attributes

Support for the BOM *lexicon definition* template component described in Section 6 is provided within each of the template component elements using a *semantics* sub-element.

The remainder of this section describes the individual schema components for each major template component element.

7.1 Model Identification

The *model identification* template component associated with the Model Identification category provides a mechanism for identifying the essential metadata of a BOM. The *model identification* template component is identified as the *modelIdentification* element in the BOM DIF Schema (see Figure 7.1) and is defined by the *modelIdentificationType*. The elements used to represent the *modelIdentificationType* for the metadata model are depicted in Figure 7-3.

Figure 7-3 *modelIdentification* Elements

The *keyword* element identified as part of the *modelIdentification* is represented as a complex data type as depicted in Figure 7-4.

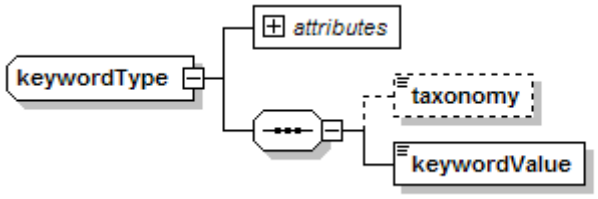


Figure 7-4 *keyword* Element

The *poc* element identified as part of the *modelIdentification* is represented as a complex data type as depicted in Figure 7-5.

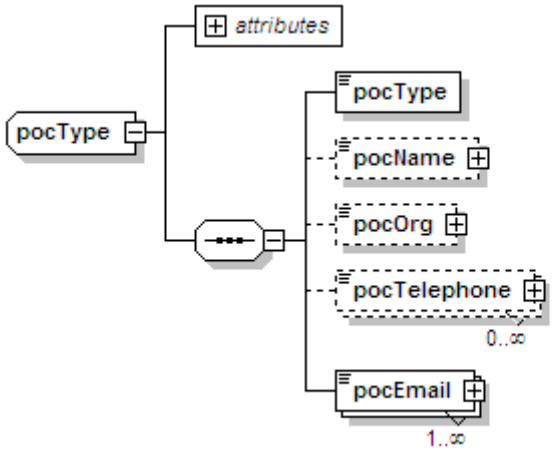


Figure 7-5 *poc* Element

The *reference* element identified as part of the *modelIdentification* is represented as a complex data type as depicted in Figure 7-6.

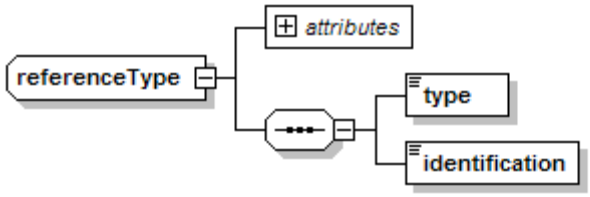


Figure 7-6 *reference* Element

The BOM schema allows a *glyph* to be identified for the purposes of representing an image depicting the BOM that can be represented symbolically on a tool palette, database, or a web page. The attributes for the *glyph* data type are found in Figure 7-7.

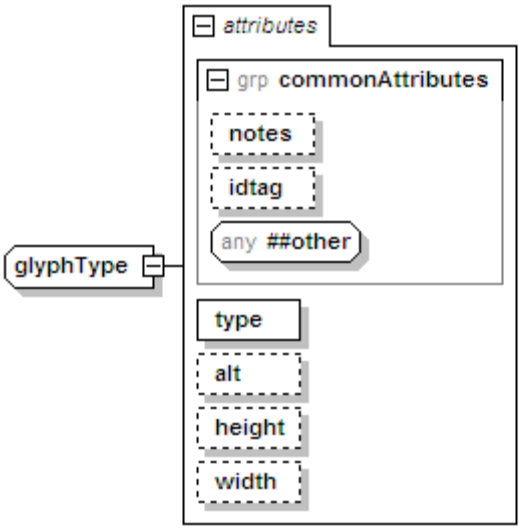


Figure 7-7 *glyph* Attributes

7.2 Conceptual Model Definition

The Conceptual Model Definition category provides a mechanism for identifying *patterns of interplay*, *state machines*, *entity types,* and *event types* within a BOM.

7.2.1 Pattern of Interplay

The *pattern of interplay* template component provides a mechanism for identifying *pattern actions* (including *variations* and *exceptions*) necessary for fulfilling the *pattern of interplay* being represented by the BOM. The *pattern of interplay* template component is identified as the *patternInterplay* element in the BOM DIF Schema (see Figure 7.1) and is defined by the *patternInterplayType*. The elements used to represent the *patternInterplayType* are depicted in Figure 7-8.

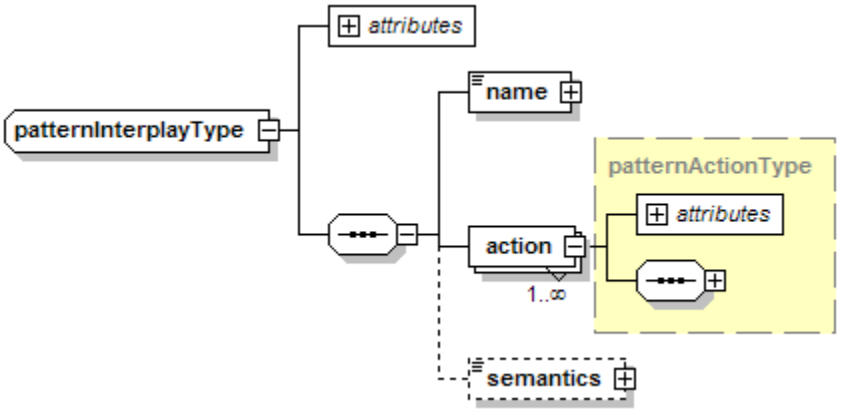


Figure 7-8 BOM *patternInterplayType* Elements

The *pattern of interplay* describing a BOM can be represented by one or more *pattern actions*. Each *pattern action* can also include an *exception*, which is an undesired action that might occur, and a *variation*, which identifies a different way for the *pattern action* to be accomplished. The *pattern actions*, *variations,* and

exceptions can be associated with a defined *event type* within the BOM, or reference a completely unique *bom*. The *senders* and *receivers* are associated with a defined *entity type* within the BOM.

The *pattern action* is identified as the *patternAction* element in Figure 7-8 and is defined by the *patternActionType*. The elements used to represent the *patternActionType* are depicted in Figure 7-9.

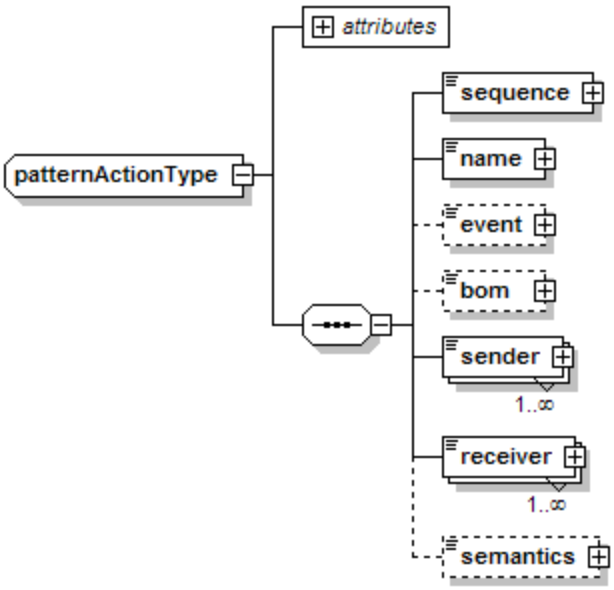


Figure 7-9 BOM *patternAction* Elements

The *exception* and *variation* are identified as the *exception* and *variation* elements in Figure 7-8 and are defined by the *exceptionType* and *variationType* respectively. The elements used to represent the *exceptionType* are depicted in Figure 7-10, and the elements used to represent the *variationType* are depicted in Figure 7-11.

A reference to a BOM *event* carrying out a *pattern action*, *exception,* or *variation* is defined by the *event type* (see Section 7.2.4). Otherwise, the *pattern action*, *exception,* or *variation* can be supported by a unique BOM using the *bom* element. The *sender* and *receiver* elements provide a means to reference an *entity* type (see Section 7.2.3).

It should be noted that an *exception* requires a condition to be met, which is an added element not supported by the *action*, and is optional for a *variation* element.

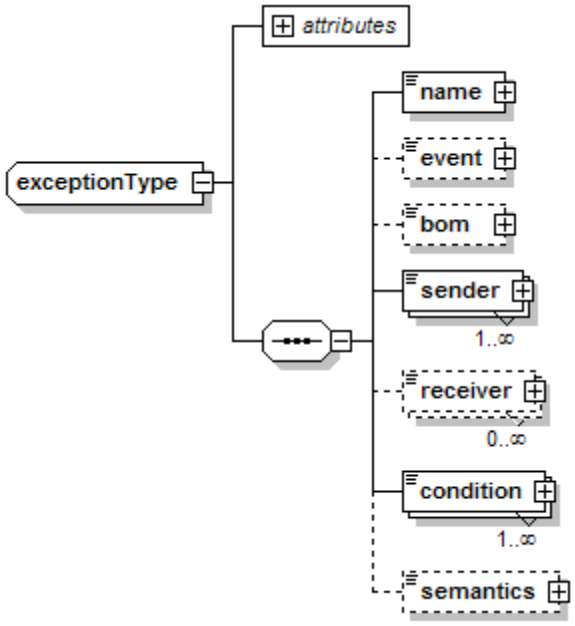


Figure 7-10 BOM *exceptionType* Elements

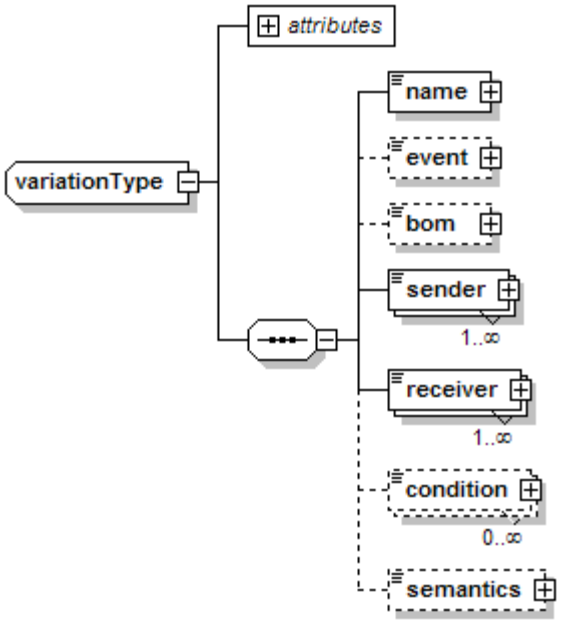


Figure 7-11 BOM *variationType* Elements

7.2.2 State Machine

The *state machine* template component provides a mechanism for identifying *states* of a *conceptual entity* to support one or more *patterns of interplay*. The *state machine* template component is identified as the *stateMachine* element in the BOM DIF Schema (see Figure 7.1) and is defined by the *stateMachineType*. The elements used to represent a *stateMachineType* are depicted in Figure 7-12.

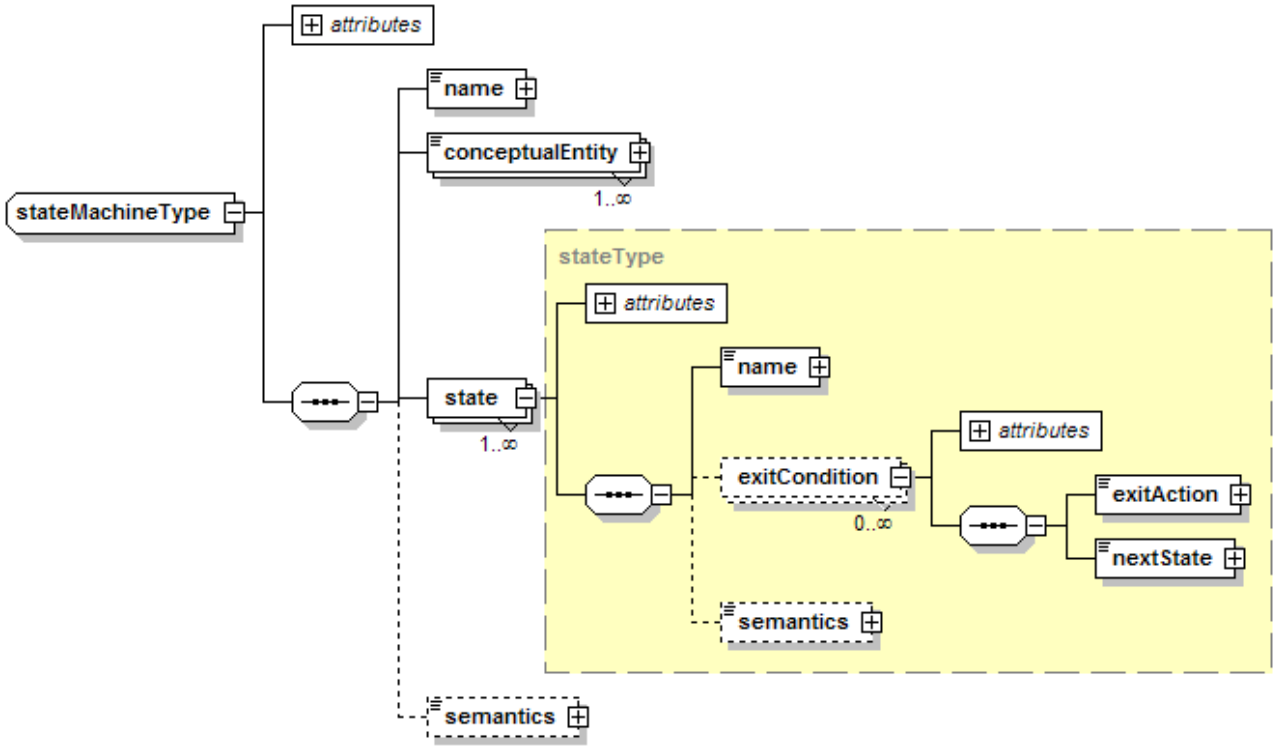


Figure 7-12 BOM *stateMachine* Elements

Notice that each *state* within the *state machine* of a BOM can contain multiple *exitConditions* to transition from the *state*. Also these conditions are linked with an *exitAction*, which is defined in a BOM *pattern of Interplay* (see Section 7.2.1), and reference the *nextState* that succeeds the current *state* when the *exitAction* occurs.

7.2.3 BOM Entity Types

The *entity type* template component provides a mechanism for identifying conceptual entities. The *entity type* template component is identified as the *entityTypes* element in the BOM DIF Schema (see Figure 7.1) and is defined by the *entityType*. The elements used to represent an *entityType* are depicted in Figure 7-13.

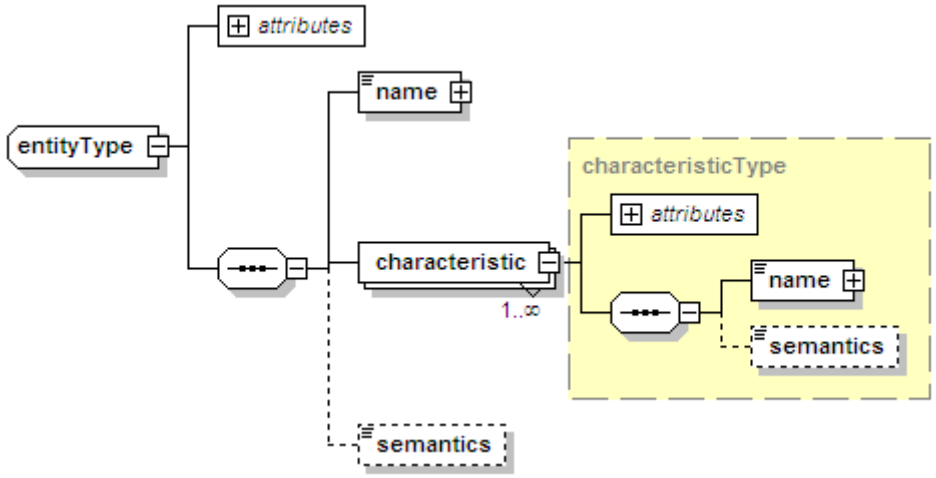


Figure 7-13 BOM *entityType* Element

7.2.4 BOM Event Types

The *event type* template component provides a mechanism for identifying conceptual events. The *event type* template component is identified as the *eventTypes* element in the BOM DIF Schema Schema (see Figure 7.1) and is defined by the *eventType*. The elements used to represent an *eventType* are depicted in Figure 7-14.

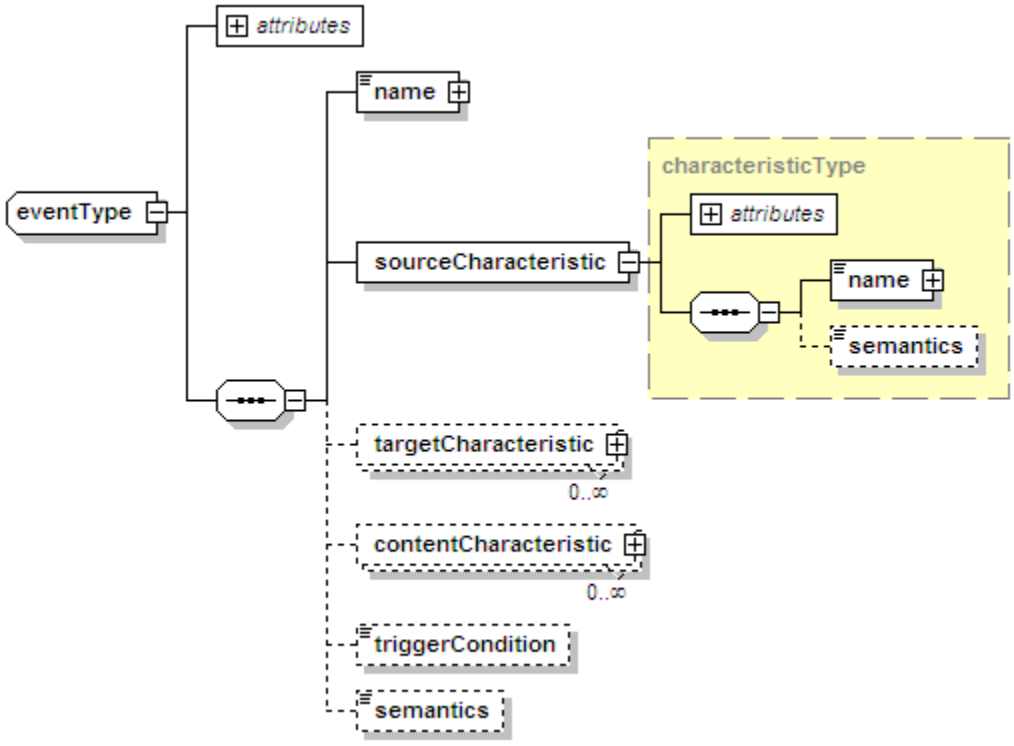


Figure 7-14 BOM *eventType* Element

One or more *triggers* or *messages* can be chosen for representing the *events* pertaining to the *pattern of interplay* that a BOM is intended to characterize. A *trigger* represents an undirected simulation *event* without any information about intended *receiver(s)*. However, a *message* represents a directed simulation *event* including information on the intended *receiver*.

7.3 Model Mapping

The Model Mapping category provides a mechanism for identifying *entity mappings*, and *event mappings* within a BOM. The *entity mappings* and *event mappings* template components are contained by the *modelMapping* element in the BOM DIF Schema (see Figure 7.1)

7.3.1 Entity Type Mappings

The *entity mappings* are defined by the *entityTypeMappingsType*. The elements used to represent the *entityTypeMappingsType* are depicted in Figure 7-15.

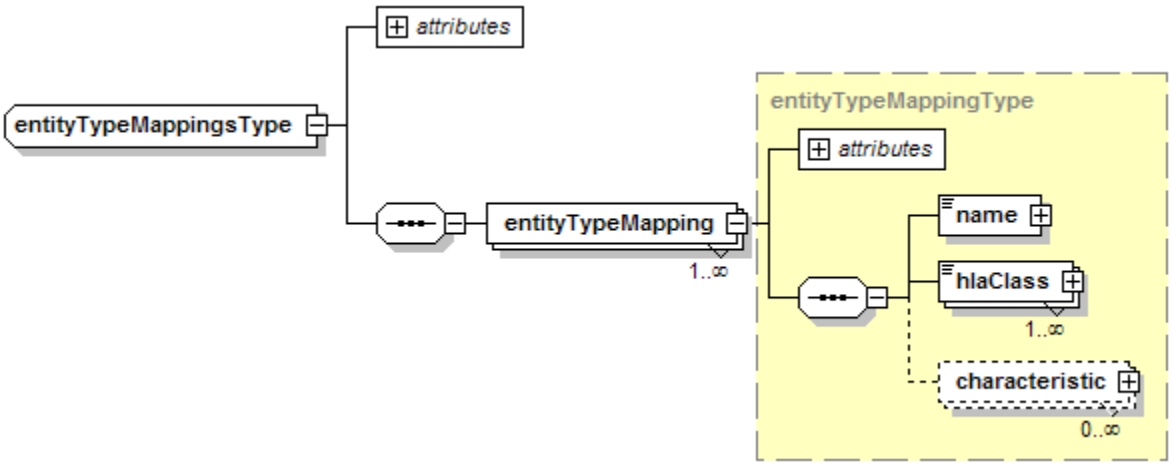


Figure 7-15 BOM *entityTypeMappingsType* Elements

The *characteristic* element identified as part of the *entityTypeMappingType* is represented using the complex type depicted in Figure 7-16.

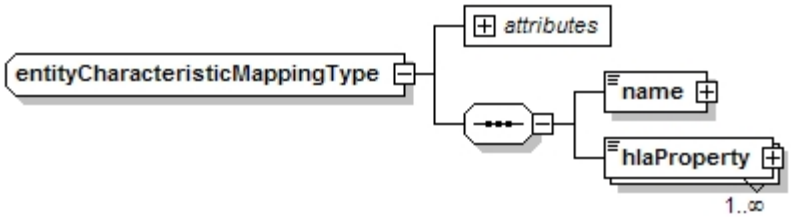


Figure 7-16 BOM *entityCharacteristicMappingType* Elements

7.3.2 Event Type Mappings

The *event mappings* are defined by the *eventTypeMappingsType*. The elements used to represent the *eventTypeMappingsType* are depicted in Figure 7-17.

The *sourceCharacteristic, targetCharacteristic, contentCharacteristic, and triggerCondition* elements identified as part of the *eventTypeMappingType* are represented using the complex type depicted in Figure 7-18.

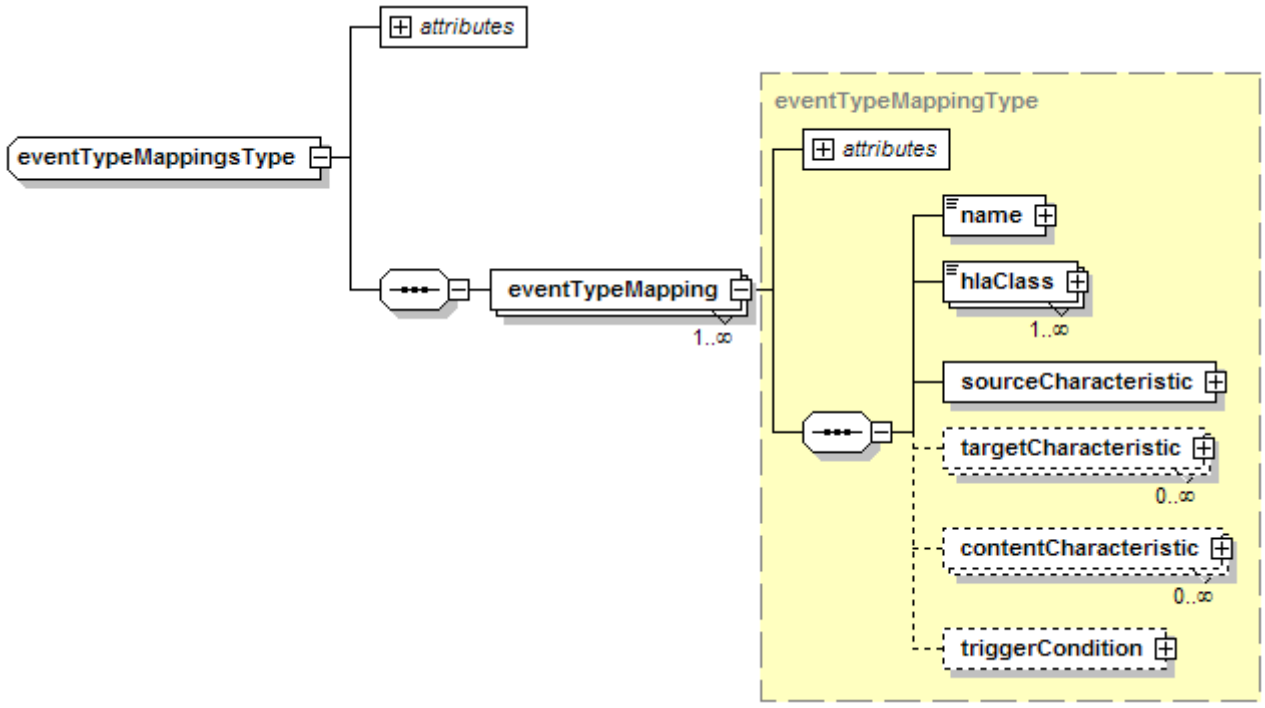


Figure 7-17 BOM *eventTypeMappingsType* Elements

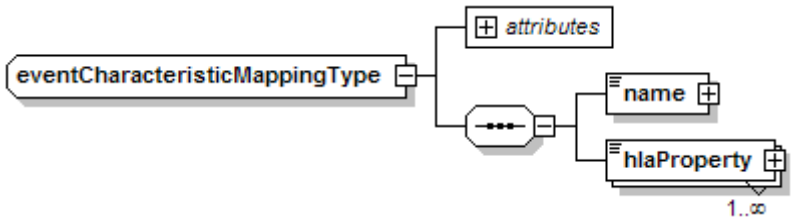


Figure 7-18 BOM *eventCharacteristicMappingType* Elements

7.4 Object Model Definition

The Object Model Definition category provides a mechanism for identifying the class structure elements necessary to support the capabilities described in the *conceptual model* of a BOM. An excerpt of the HLA OMT Specification is used to identify what is needed for representing the class structure elements in terms of *HLA object classes* and *interaction classes*, *attributes*, *parameters,* and *datatypes* template components. These template components for representing the Object Model Definition category are supported by the *objectModelDefintion* element in the BOM DIF Schema (see Figure 7.1) and are defined by the *objectModelDefinitionType*. The elements used to represent the *objectModelDefinitionType* of a BOM are depicted in Figure 7-19.

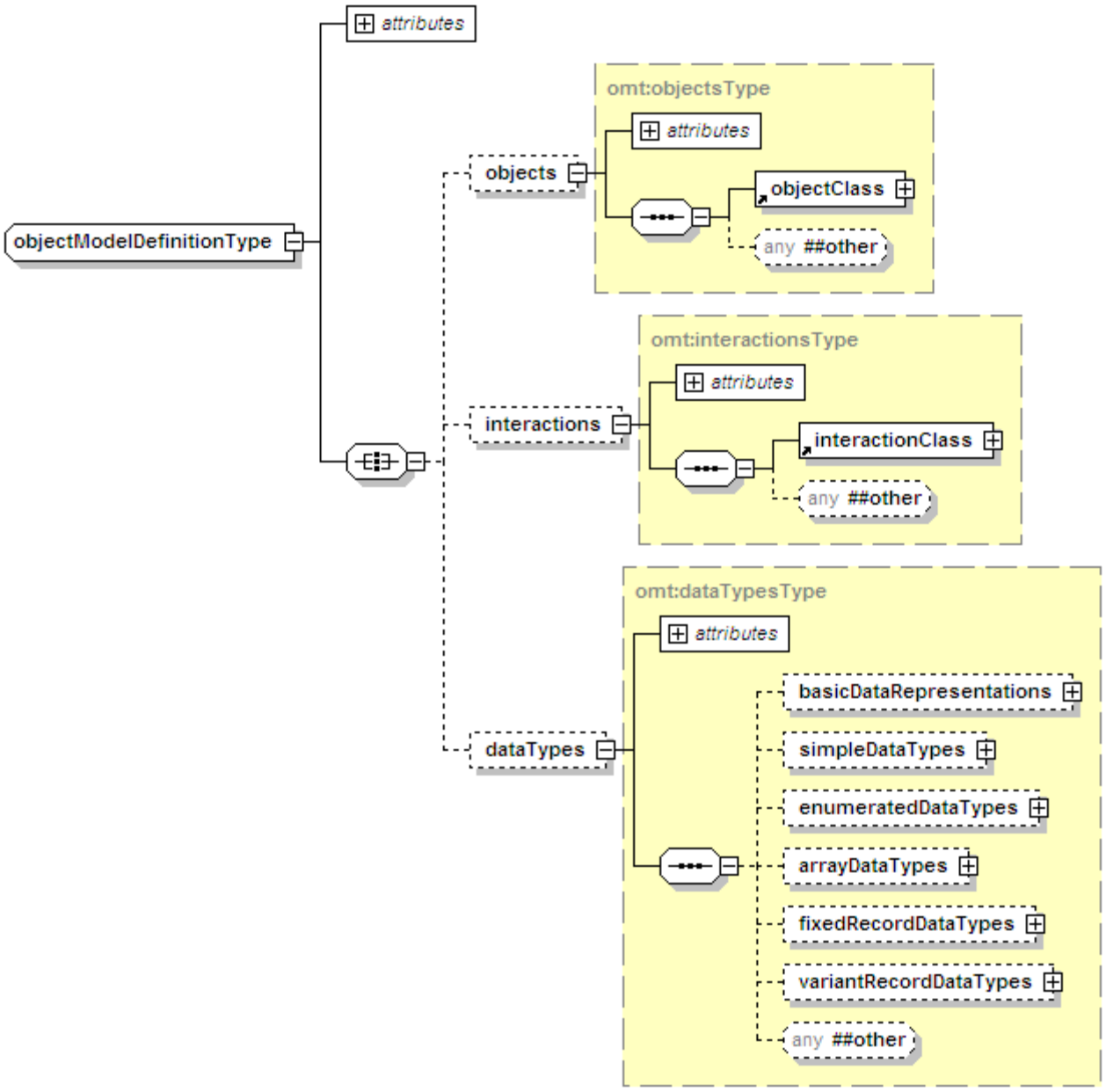


Figure 7-19 BOM *objectModelDefinitionType E*lements

Not leveraged from the original HLA OMT Specification are the federate/federation connections related to technical interoperability. These template components have been considered a deeper level of detail not needed for representing simulation *patterns of interplay* and components. The HLA OMT components of interest for a BOM include the following:

- *object classes / attributes*
- *interaction classes / parameters*
- *dataTypes*, which include
  - *basicDataRepresentations*
  - *simpleDataTypes*
  - *enumeratedDataTypes*

- *arrayDataTypes*
- *fixedRecordDataTypes*
- *variantRecordDataTypes*

The HLA OMT components not used in a BOM include the following:

- *dimensions*
- *time*
- *tags*
- *synchronizations*
- *transportations*
- *switches*

7.5 Notes

The *notes* template component, which is leveraged from the HLA OMT Specification, provides a mechanism for associating *notes* with any of the other BOM template components. Within the BOM DIF, any XML Element can be supported by a *notes reference*. The actual *note* to the *notes reference* would be contained in the *notes* template component. The *notes* template component is identified as the *notes* element in the BOM DIF Schema (see Figure 7.1) and is defined by the *notesType*. The elements used to represent the *notesType* are depicted in Figure 7-20.

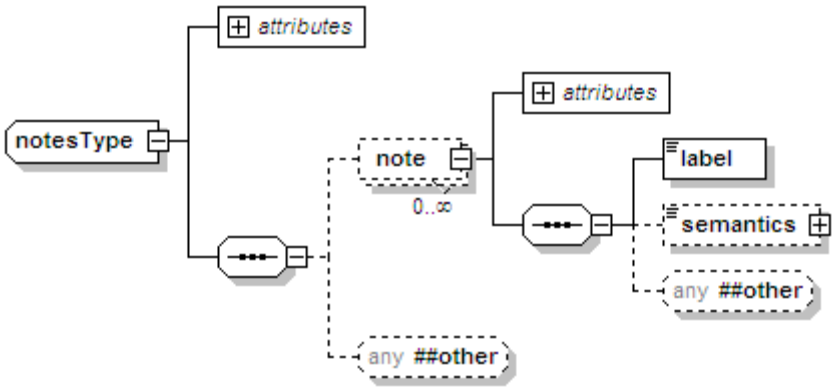


Figure 7-20 BOM *notesType* Elements

This page left intentionally blank

Annex A – BOM Schemas

Listings of the following BOM DIF related-schemas are provided in this Annex:

- **BOM Schema** - The core schema used to document a BOM. The file name for this schema is identified as **BOM_v2006.xsd**.
- **Model ID Schema** – The schema used to document the model identification metadata identified within a BOM or other models such as future FOMs and SOMs. The file name for this schema is identified as **ModelID_v2006.xsd** and is recognized by the **modelID** namespace.
- **HLA OMT Schema** – The schema used to document the class structures of a BOM using HLA OMT constructs. The file name for this schema is identified as **IEEE1516.2-2006-D2v0.82.xsd** and is recognized by the **IEEE1516.2-2006** namespace.

7.6 BOM Schema

The following listing provides the core XML schema representing the BOM DIF. This listing of the BOM schema, identified as **BOM_v2006.xsd**, imports the **modelID** namespace and schema for representing metadata, and uses the HLA OMT **IEEE1516.2-2006** namespace and schema for representing the interface elements of the Object Model Definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2006 U (http://www.altova.com) by Tram Chase (SimVentions) -->
<!-- created with XML Spy v4.2 U (http://www.xmlspy.com) by Björn Löfstrand (Pitch Kunskapsutveckling AB) -->
<!--W3C Schema developed by the SISO BOM Product Develop Group (PDG) - version 2006 - copyright 2006 -->
<!--Schema for BOM DIF

Version 0.11 - June 29, 2005
Version 0.12 - Sep 8, 2005
Version 2006 - March 25, 2005

-->
<xs:schema xmlns="http://www.sisostds.org/schemas/bom" xmlns:omt="http://www.sisostds.org/schemas/IEEE1516.2-2006"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:modelID="http://www.sisostds.org/schemas/modelID"
targetNamespace="http://www.sisostds.org/schemas/bom" elementFormDefault="qualified">
	<xs:import namespace="http://www.sisostds.org/schemas/IEEE1516.2-2006" schemaLocation="IEEE1516.2-2006-D2v0.82.xsd"/>
	<xs:import namespace="http://www.sisostds.org/schemas/modelID" schemaLocation="ModelID_v2006.xsd"/>
	<xs:annotation>
		<xs:documentation xml:lang="en">Schema for BOM DIF</xs:documentation>
	</xs:annotation>
	<xs:element name="BOM" type="BOMType"/>
	<xs:simpleType name="blName">
		<xs:restriction base="xs:string">
			<xs:pattern value="\S*"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:complexType name="patternInterplayType" id="patternInterplay">
		<xs:sequence>
			<xs:element name="name" type="modelID:IdentifierType"/>
			<xs:element name="action" maxOccurs="unbounded">
				<xs:complexType>
					<xs:complexContent>
						<xs:extension base="patternActionType">
							<xs:choice minOccurs="0" maxOccurs="unbounded">
								<xs:element name="exception" type="exceptionType"/>
								<xs:element name="variation" type="variationType"/>
							</xs:choice>
						</xs:extension>
					</xs:complexContent>
				</xs:complexType>
			</xs:element>
			<xs:element name="semantics" type="modelID:String" minOccurs="0"/>
		</xs:sequence>
		<xs:attributeGroup ref="modelID:commonAttributes"/>
	</xs:complexType>
	<xs:complexType name="patternActionType">
		<xs:sequence>
			<xs:element name="sequence">
				<xs:complexType>
					<xs:simpleContent>
						<xs:extension base="blName">
```

```
                                                <xs:attributeGroup ref="modelID:commonAttributes"/>
                                        </xs:extension>
                                </xs:simpleContent>
                        </xs:complexType>
                </xs:element>
                <xs:element name="name" type="modelID:IdentifierType"/>
                <xs:element name="event" minOccurs="0">
                        <xs:complexType>
                                <xs:simpleContent>
                                        <xs:extension base="xs:anyURI">
                                                <xs:attributeGroup ref="modelID:commonAttributes"/>
                                        </xs:extension>
                                </xs:simpleContent>
                        </xs:complexType>
                </xs:element>
                <xs:element name="bom" minOccurs="0">
                        <xs:complexType>
                                <xs:simpleContent>
                                        <xs:extension base="xs:anyURI">
                                                <xs:attributeGroup ref="modelID:commonAttributes"/>
                                        </xs:extension>
                                </xs:simpleContent>
                        </xs:complexType>
                </xs:element>
                <xs:element name="sender" type="modelID:NonEmptyString" maxOccurs="unbounded"/>
                <xs:element name="receiver" type="modelID:NonEmptyString" maxOccurs="unbounded"/>
                <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
        </xs:sequence>
        <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="exceptionType">
        <xs:sequence>
                <xs:element name="name" type="modelID:IdentifierType"/>
                <xs:element name="event" minOccurs="0">
                        <xs:complexType>
                                <xs:simpleContent>
                                        <xs:extension base="xs:anyURI">
                                                <xs:attributeGroup ref="modelID:commonAttributes"/>
                                        </xs:extension>
                                </xs:simpleContent>
                        </xs:complexType>
                </xs:element>
                <xs:element name="bom" minOccurs="0">
                        <xs:complexType>
                                <xs:simpleContent>
                                        <xs:extension base="xs:anyURI">
                                                <xs:attributeGroup ref="modelID:commonAttributes"/>
                                        </xs:extension>
                                </xs:simpleContent>
                        </xs:complexType>
                </xs:element>
                <xs:element name="sender" type="modelID:NonEmptyString" maxOccurs="unbounded"/>
                <xs:element name="receiver" type="modelID:NonEmptyString" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="condition" type="modelID:NonEmptyString" maxOccurs="unbounded"/>
                <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
        </xs:sequence>
        <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="variationType">
        <xs:sequence>
                <xs:element name="name" type="modelID:IdentifierType"/>
                <xs:element name="event" minOccurs="0">
                        <xs:complexType>
                                <xs:simpleContent>
                                        <xs:extension base="xs:anyURI">
                                                <xs:attributeGroup ref="modelID:commonAttributes"/>
                                        </xs:extension>
                                </xs:simpleContent>
                        </xs:complexType>
                </xs:element>
                <xs:element name="bom" minOccurs="0">
                        <xs:complexType>
                                <xs:simpleContent>
                                        <xs:extension base="xs:anyURI">
                                                <xs:attributeGroup ref="modelID:commonAttributes"/>
                                        </xs:extension>
                                </xs:simpleContent>
                        </xs:complexType>
                </xs:element>
```

```
                    <xs:element name="sender" type="modelID:NonEmptyString" maxOccurs="unbounded"/>
                    <xs:element name="receiver" type="modelID:NonEmptyString" maxOccurs="unbounded"/>
                    <xs:element name="condition" type="modelID:NonEmptyString" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="stateType">
            <xs:sequence>
                    <xs:element name="name" type="modelID:IdentifierType"/>
                    <xs:element name="exitCondition" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                                    <xs:sequence>
                                            <xs:element name="exitAction" type="modelID:NonEmptyString"/>
                                            <xs:element name="nextState" type="modelID:NonEmptyString"/>
                                    </xs:sequence>
                                    <xs:attributeGroup ref="modelID:commonAttributes"/>
                            </xs:complexType>
                    </xs:element>
                    <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="stateMachineType">
            <xs:sequence>
                    <xs:element name="name" type="modelID:IdentifierType"/>
                    <xs:element name="conceptualEntity" maxOccurs="unbounded">
                            <xs:complexType>
                                    <xs:simpleContent>
                                            <xs:extension base="xs:anyURI">
                                                    <xs:attributeGroup ref="modelID:commonAttributes"/>
                                            </xs:extension>
                                    </xs:simpleContent>
                            </xs:complexType>
                    </xs:element>
                    <xs:element name="state" maxOccurs="unbounded">
                            <xs:complexType>
                                    <xs:complexContent>
                                            <xs:extension base="stateType"/>
                                    </xs:complexContent>
                            </xs:complexType>
                    </xs:element>
                    <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="objectModelDefinitionType">
            <xs:all>
                    <xs:element name="objects" type="omt:objectsType" minOccurs="0"/>
                    <xs:element name="interactions" type="omt:interactionsType" minOccurs="0"/>
                    <xs:element name="dataTypes" type="omt:dataTypesType" minOccurs="0"/>
            </xs:all>
            <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="characteristicType">
            <xs:sequence>
                    <xs:element name="name" type="modelID:IdentifierType"/>
                    <xs:element name="semantics" type="modelID:String" minOccurs="0">
                            <xs:annotation>
                                    <xs:documentation>lexicon entry for this characteristic</xs:documentation>
                            </xs:annotation>
                    </xs:element>
            </xs:sequence>
            <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="entityCharacteristicMappingType">
            <xs:sequence>
                    <xs:element name="name" type="modelID:IdentifierType"/>
                    <xs:element name="hlaProperty" type="modelID:NonEmptyString" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="eventCharacteristicMappingType">
            <xs:sequence>
                    <xs:element name="name" type="modelID:IdentifierType"/>
                    <xs:element name="hlaProperty" type="modelID:NonEmptyString" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
```

```
<xs:complexType name="entityType">
	<xs:sequence>
		<xs:element name="name" type="modelID:IdentifierType"/>
		<xs:element name="characteristic" type="characteristicType" maxOccurs="unbounded"/>
		<xs:element name="semantics" type="modelID:String" minOccurs="0">
			<xs:annotation>
				<xs:documentation>lexicon entry for this entity type</xs:documentation>
			</xs:annotation>
		</xs:element>
	</xs:sequence>
	<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="eventType">
	<xs:sequence>
		<xs:element name="name" type="modelID:IdentifierType"/>
		<xs:element name="sourceCharacteristic">
			<xs:complexType>
				<xs:complexContent>
					<xs:extension base="characteristicType"/>
				</xs:complexContent>
			</xs:complexType>
		</xs:element>
		<xs:element name="targetCharacteristic" type="characteristicType" minOccurs="0" maxOccurs="unbounded"/>
		<xs:element name="contentCharacteristic" type="characteristicType" minOccurs="0" maxOccurs="unbounded"/>
		<xs:element name="triggerCondition" type="modelID:NonEmptyString" minOccurs="0"/>
		<xs:element name="semantics" type="modelID:String" minOccurs="0">
			<xs:annotation>
				<xs:documentation>lexicon entry for this event type</xs:documentation>
			</xs:annotation>
		</xs:element>
	</xs:sequence>
	<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="entityTypeMappingsType">
	<xs:sequence>
		<xs:element name="entityTypeMapping" type="entityTypeMappingType" maxOccurs="unbounded"/>
	</xs:sequence>
	<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="eventTypeMappingsType">
	<xs:sequence>
		<xs:element name="eventTypeMapping" type="eventTypeMappingType" maxOccurs="unbounded"/>
	</xs:sequence>
	<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="entityTypeMappingType">
	<xs:sequence>
		<xs:element name="name" type="modelID:IdentifierType"/>
		<xs:element name="hlaClass" type="modelID:NonEmptyString" maxOccurs="unbounded"/>
		<xs:element name="characteristic" type="entityCharacteristicMappingType" minOccurs="0" maxOccurs="unbounded"/>
	</xs:sequence>
	<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="eventTypeMappingType">
	<xs:sequence>
		<xs:element name="name" type="modelID:IdentifierType"/>
		<xs:element name="hlaClass" type="modelID:NonEmptyString" maxOccurs="unbounded"/>
		<xs:element name="sourceCharacteristic" type="eventCharacteristicMappingType"/>
		<xs:element name="targetCharacteristic" type="eventCharacteristicMappingType" minOccurs="0"
maxOccurs="unbounded"/>
		<xs:element name="contentCharacteristic" type="eventCharacteristicMappingType" minOccurs="0"
maxOccurs="unbounded"/>
		<xs:element name="triggerCondition" type="eventCharacteristicMappingType" minOccurs="0"/>
	</xs:sequence>
	<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="BOMType">
	<xs:sequence>
		<xs:element name="modelIdentification" type="modelID:modelIdentificationType"/>
		<xs:element name="conceptualModelDefinition" type="conceptualModelType" minOccurs="0"/>
		<xs:element name="modelMapping" type="modelMappingType" minOccurs="0"/>
		<xs:element name="objectModelDefinition" type="objectModelDefinitionType" minOccurs="0"/>
		<xs:element name="notes" minOccurs="0">
			<xs:complexType>
				<xs:complexContent>
					<xs:extension base="omt:notesType"/>
				</xs:complexContent>
			</xs:complexType>
		</xs:element>
```

```
                </xs:sequence>
                <xs:attributeGroup ref="modelID:commonAttributes"/>
        </xs:complexType>
        <xs:complexType name="modelMappingType">
                <xs:sequence>
                        <xs:element name="entityTypeMappings" type="entityTypeMappingsType" minOccurs="0"/>
                        <xs:element name="eventTypeMappings" type="eventTypeMappingsType" minOccurs="0"/>
                </xs:sequence>
                <xs:attributeGroup ref="modelID:commonAttributes"/>
        </xs:complexType>
        <xs:complexType name="conceptualModelType">
                <xs:sequence>
                        <xs:element name="patternInterplay" type="patternInterplayType" id="patternID" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element name="stateMachine" type="stateMachineType" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="entityTypes" minOccurs="0">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="entityType" type="entityType" maxOccurs="unbounded"/>
                                        </xs:sequence>
                                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                                </xs:complexType>
                        </xs:element>
                        <xs:element name="eventTypes" minOccurs="0">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="eventType" type="eventType" minOccurs="0"
maxOccurs="unbounded"/>
                                        </xs:sequence>
                                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
                <xs:attributeGroup ref="modelID:commonAttributes"/>
        </xs:complexType>
</xs:schema>
```

This page left intentionally blank

7.7 Model Identification Schema

The following listing provides the XML schema for representing model identification metadata within a BOM. The Model Identification schema, identified as **ModelID_v2006.xsd,** is recognized by the **modelID** namespace. Providing it as a separate namespace and schema makes it available for inclusion by other XML-based DIFs, which may require common metadata representation. The following listing provides the schema defining this **modelID** namespace.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2006 U (http://www.altova.com) by Tram Chase (SimVentions) -->
<!-- created with XMLSpy v2006 U (http://www.altova.com) by Tram Chase (SimVentions) -->
<!--W3C Schema developed by the SISO BOM Product Develop Group (PDG) - version 2006 - copyright 2006 -->
<!--Schema for BOM DIF

    Version 0.12 - Sep 8, 2005
    Version 2006 - March 25, 2005
-->
<xs:schema xmlns="http://www.sisostds.org/schemas/modelID" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.sisostds.org/schemas/modelID" elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:complexType name="String">
                <xs:simpleContent>
                        <xs:extension base="xs:string">
                                <xs:attributeGroup ref="commonAttributes"/>
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>
        <xs:complexType name="IdentifierType">
                <xs:simpleContent>
                        <xs:extension base="xs:NCName">
                                <xs:attributeGroup ref="commonAttributes"/>
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>
        <xs:complexType name="NonEmptyString">
                <xs:simpleContent>
                        <xs:extension base="nonEmptyString">
                                <xs:attributeGroup ref="commonAttributes"/>
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>
        <xs:simpleType name="OMTypeEnumerations">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="FOM"/>
                        <xs:enumeration value="SOM"/>
                        <xs:enumeration value="BOM"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="OMTypeUnion">
                <xs:union memberTypes="OMTypeEnumerations nonEmptyString"/>
                <!--this allows enumerations to be used plus any user defined content -->
        </xs:simpleType>
        <xs:simpleType name="glyphTypeEnumerations">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="BITMAP"/>
                        <xs:enumeration value="JPG"/>
                        <xs:enumeration value="GIF"/>
                        <xs:enumeration value="PNG"/>
                        <xs:enumeration value="TIFF"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="glyphTypeUnion">
                <xs:union memberTypes="glyphTypeEnumerations xs:string"/>
                <!--this allows enumerations to be used plus any user defined content -->
        </xs:simpleType>
        <xs:complexType name="glyphType" mixed="true">
                <xs:simpleContent>
                        <xs:extension base="xs:base64Binary">
                                <xs:attributeGroup ref="commonAttributes"/>
                                <xs:attribute name="type" type="glyphTypeUnion" use="required"/>
                                <xs:attribute name="height" type="xs:short"/>
                                <xs:attribute name="width" type="xs:short"/>
                                <xs:attribute name="alt" type="xs:string"/>
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>
```

```
<xs:complexType name="pocType">
	<xs:sequence>
		<xs:element name="pocType" type="pocTypeType">
			<xs:annotation>
				<xs:documentation>specifies the role that the POC has withto the model</xs:documentation>
			</xs:annotation>
		</xs:element>
		<xs:element name="pocName" type="String" minOccurs="0"/>
		<xs:element name="pocOrg" type="String" minOccurs="0"/>
		<xs:element name="pocTelephone" type="String" minOccurs="0" maxOccurs="unbounded"/>
		<xs:element name="pocEmail" type="String" maxOccurs="unbounded"/>
	</xs:sequence>
</xs:complexType>
<xs:attributeGroup name="commonAttributes">
	<!--this is the common attributes for any element -->
	<xs:attribute name="notes" type="xs:IDREFS" use="optional"/>
	<xs:attribute name="idtag" type="xs:ID" use="optional"/>
	<xs:anyAttribute namespace="##other"/>
</xs:attributeGroup>
<xs:simpleType name="nonEmptyString">
	<xs:restriction base="xs:string">
		<xs:minLength value="1"/>
	</xs:restriction>
</xs:simpleType>
<xs:simpleType name="SecurityClassificationEnumeration">
	<xs:restriction base="xs:string">
		<xs:enumeration value="Unclassified"/>
		<xs:enumeration value="Confidential"/>
		<xs:enumeration value="Secret"/>
		<xs:enumeration value="Top Secret"/>
	</xs:restriction>
</xs:simpleType>
<xs:simpleType name="SecurityClassificationUnion">
	<xs:union memberTypes="SecurityClassificationEnumeration nonEmptyString"/>
</xs:simpleType>
<xs:complexType name="modelType">
	<xs:simpleContent>
		<xs:extension base="OMTypeUnion">
			<xs:attributeGroup ref="commonAttributes"/>
		</xs:extension>
	</xs:simpleContent>
</xs:complexType>
<xs:complexType name="securityClassificationType">
	<xs:simpleContent>
		<xs:extension base="SecurityClassificationUnion">
			<xs:attributeGroup ref="commonAttributes"/>
		</xs:extension>
	</xs:simpleContent>
</xs:complexType>
<xs:simpleType name="ApplicationDomainEnumerations">
	<xs:restriction base="xs:string">
		<xs:enumeration value="Analysis"/>
		<xs:enumeration value="Training"/>
		<xs:enumeration value="Test and Evaluation"/>
		<xs:enumeration value="Engineering"/>
		<xs:enumeration value="Acquisition"/>
	</xs:restriction>
</xs:simpleType>
<xs:simpleType name="ApplicationDomainUnion">
	<xs:union memberTypes="ApplicationDomainEnumerations xs:string"/>
</xs:simpleType>
<xs:complexType name="applicationDomainType">
	<xs:simpleContent>
		<xs:extension base="ApplicationDomainUnion">
			<xs:attributeGroup ref="commonAttributes"/>
		</xs:extension>
	</xs:simpleContent>
</xs:complexType>
<xs:simpleType name="POCTypeEnumeration">
	<xs:restriction base="xs:string">
		<xs:enumeration value="Primary author"/>
		<xs:enumeration value="Contributor"/>
		<xs:enumeration value="Proponent"/>
		<xs:enumeration value="Sponsor"/>
		<xs:enumeration value="Release authority"/>
		<xs:enumeration value="Technical POC"/>
	</xs:restriction>
</xs:simpleType>
<xs:simpleType name="POCTypeUnion">
```

```
                <xs:union memberTypes="POCTypeEnumeration nonEmptyString"/>
        </xs:simpleType>
        <xs:complexType name="pocTypeType">
                <xs:simpleContent>
                        <xs:extension base="POCTypeUnion">
                                <xs:attributeGroup ref="commonAttributes"/>
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>
        <xs:simpleType name="referenceTypeEnumerations">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Source Material"/>
                        <xs:enumeration value="Conceptual Model"/>
                        <xs:enumeration value="Related BOM"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="referenceTypeUnion">
                <xs:union memberTypes="referenceTypeEnumerations xs:string"/>
                <!--this allows enumerations to be used plus any user defined content -->
        </xs:simpleType>
        <xs:complexType name="referenceType">
                <xs:sequence>
                        <xs:element name="type">
                                <xs:complexType>
                                        <xs:simpleContent>
                                                <xs:extension base="referenceTypeUnion">
                                                        <xs:attributeGroup ref="commonAttributes"/>
                                                </xs:extension>
                                        </xs:simpleContent>
                                </xs:complexType>
                        </xs:element>
                        <xs:element name="identification">
                                <xs:complexType>
                                        <xs:simpleContent>
                                                <xs:extension base="xs:anyURI">
                                                        <xs:attributeGroup ref="commonAttributes"/>
                                                </xs:extension>
                                        </xs:simpleContent>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
                <xs:attributeGroup ref="commonAttributes"/>
        </xs:complexType>
        <xs:complexType name="keywordType">
                <xs:sequence>
                        <xs:element name="taxonomy" type="String" minOccurs="0"/>
                        <xs:element name="keywordValue" type="NonEmptyString"/>
                </xs:sequence>
                <xs:attributeGroup ref="commonAttributes"/>
        </xs:complexType>
        <xs:complexType name="modelIdentificationType">
                <xs:sequence>
                        <xs:element name="name" type="IdentifierType">
                                <xs:annotation>
                                        <xs:documentation>specifies the name assigned to the object model</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="type" type="modelType">
                                <xs:annotation>
                                        <xs:documentation>specify the type of model that is represented</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="version" type="NonEmptyString">
                                <xs:annotation>
                                        <xs:documentation>specifies the version identification assigned to the object
model</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="modificationDate" nillable="false">
                                <xs:annotation>
                                        <xs:documentation>specifies the latest date on which this version of the object model was
created or modified. The modification date shall be specified in the format "YYYY-MM-DD"</xs:documentation>
                                </xs:annotation>
                                <xs:complexType>
                                        <xs:simpleContent>
                                                <xs:extension base="xs:date">
                                                        <xs:attributeGroup ref="commonAttributes"/>
                                                </xs:extension>
                                        </xs:simpleContent>
```

```
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="securityClassification" type="securityClassificationType">
                        <xs:annotation>
                            <xs:documentation>specifies the security classification of themodel</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="releaseRestriction" type="String" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>specifies any restrictions on the release of the object models to specific
organizations or individuals</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="purpose" type="String" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>specifies the purpose for which the federate or federation was
developed</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="applicationDomain" type="applicationDomainType" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>specifies the type or class of application to which the federate or federation
applies</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="description" type="NonEmptyString"/>
                    <xs:element name="useLimitation" type="String" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>specifies any known applications for which this model has been found not to
be appropriate</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="useHistory" type="String" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>specifies a description of where this model has been
used</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="keyword" type="keywordType" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>specifies keywords that characterize the model</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="poc" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>specify an organization or awho has a particular role with respect to the
model</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:extension base="pocType">
                                    <xs:attributeGroup ref="commonAttributes"/>
                                </xs:extension>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="reference" type="referenceType" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>specifies a pointer to additional sources of information</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="other" type="String" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>specifies other data deemed relevant by the author of the object
model</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="glyph" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>specifies a glyph to visually represent the model</xs:documentation>
                        </xs:annotation>
                        <xs:complexType mixed="true">
                            <xs:simpleContent>
                                <xs:extension base="glyphType"/>
                            </xs:simpleContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:any namespace="##other" minOccurs="0"/>
                </xs:sequence>
```

```
                    <xs:attributeGroup ref="commonAttributes"/>
                </xs:complexType>
</xs:schema>
```

This page left intentionally blank

7.8 HLA OMT Schema

The following listing provides the complete HLA OMT schema based on the anticipated 2006 version of the 1516 specification. The specific HLA OMT elements used for the BOM schema include the object class, interaction class, attributes, parameters, data types, common attributes, notes, and semantics. The HLA OMT schema, identified as **IEEE1516.2-2006-D2v0.82.xsd**, which imports the **modelID** namespace and schema for representing metadata, is recognized by the **IEEE1516.2-2006** namespace. Providing it as a separate namespace and schema makes it available for inclusion by other XML-based DIFs such as the BOM Schema. The following listing provides the schema defining this **IEEE1516.2-2006** namespace.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by Tram Chase (SimVentions, Inc.) -->
<!-- created with XML Spy v4.2 U (http://www.xmlspy.com) by Björn Löfstrand (Pitch Kunskapsutveckling AB) -->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.sisostds.org/schemas/IEEE1516.2-2006"
xmlns:ns="http://www.sisostds.org/schemas/IEEE1516.2-2006" xmlns:modelID="http://www.sisostds.org/schemas/modelID"
targetNamespace="http://www.sisostds.org/schemas/IEEE1516.2-2006" elementFormDefault="qualified" attributeFormDefault="unqualified"
version="Draft2 v0.5">
        <xs:import namespace="http://www.sisostds.org/schemas/modelID" schemaLocation="ModelID_v2006.xsd"/>
        <xs:element name="objectModel" type="objectModelType">
                <xs:key name="dimensionDatatypeKey">
                        <xs:selector xpath="./ns:simpleData/ns:name | ./ns:enumeratedData/ns:name"/>
                        <xs:field xpath="."/>
                </xs:key>
                <xs:keyref name="dimensionDatatypeRef" refer="dimensionDatatypeKey">
                        <xs:selector xpath="./ns:dimensions/*/ns:dataType"/>
                        <xs:field xpath="."/>
                </xs:keyref>
                <xs:key name="representationKey">
                        <xs:annotation>
                                <xs:documentation>unique keys for identifying basic data representations</xs:documentation>
                        </xs:annotation>
                        <xs:selector xpath=".//ns:basicData"/>
                        <xs:field xpath="ns:name"/>
                </xs:key>
                <xs:keyref name="representationRef" refer="representationKey">
                        <xs:selector xpath=".//ns:representation"/>
                        <xs:field xpath="."/>
                </xs:keyref>
                <xs:key name="dataTypeKey">
                        <xs:annotation>
                                <xs:documentation>unique keys for identifying datatypes</xs:documentation>
                        </xs:annotation>
                        <xs:selector xpath="./ns:simpleData/ns:name | ./ns:enumeratedData/ns:name | .//ns:arrayData/ns:name |
.//ns:fixedRecordData/ns:name | .//ns:variantRecordData/ns:name | .//ns:note/ns:label"/>
                        <xs:field xpath="."/>
                </xs:key>
                <xs:keyref name="dataTypeRef" refer="dataTypeKey">
                        <xs:selector xpath=".//ns:dataType"/>
                        <xs:field xpath="."/>
                </xs:keyref>
                <xs:key name="dimensionKey">
                        <xs:selector xpath="./ns:dimensions/ns:dimension"/>
                        <xs:field xpath="ns:name"/>
                </xs:key>
                <xs:keyref name="dimensionRef" refer="dimensionKey">
                        <xs:selector xpath="./ns:attribute/ns:dimensions/ns:dimension | .//ns:interactionClass/ns:dimensions/ns:dimension"/>
                        <xs:field xpath="."/>
                </xs:keyref>
                <xs:key name="transportationKey">
                        <xs:selector xpath="./ns:transportations/ns:transportation"/>
                        <xs:field xpath="ns:name"/>
                </xs:key>
                <xs:keyref name="transportationRef" refer="transportationKey">
                        <xs:selector xpath=".//ns:attribute | .//ns:interactionClass"/>
                        <xs:field xpath="ns:transportation"/>
                </xs:keyref>
        </xs:element>
        <xs:element name="objectClass">
                <xs:complexType>
                        <xs:complexContent>
                                <xs:extension base="objectClassType"/>
                        </xs:complexContent>
                </xs:complexType>
```

```
                <xs:unique name="className">
                        <xs:annotation>
                                <xs:documentation>ensures uniqueness of objectClass names among class siblings</xs:documentation>
                        </xs:annotation>
                        <xs:selector xpath="./ns:objectClass"/>
                        <xs:field xpath="ns:name"/>
                </xs:unique>
                <xs:unique name="attributeName">
                        <xs:selector xpath="./ns:attribute"/>
                        <xs:field xpath="ns:name"/>
                </xs:unique>
        </xs:element>
        <xs:element name="interactionClass">
                <xs:complexType>
                        <xs:complexContent>
                                <xs:extension base="interactionClassType"/>
                        </xs:complexContent>
                </xs:complexType>
                <xs:unique name="interactionName">
                        <xs:annotation>
                                <xs:documentation>ensures uniqueness of interactionClass names among class
siblings</xs:documentation>
                        </xs:annotation>
                        <xs:selector xpath="./ns:interactionClass"/>
                        <xs:field xpath="ns:name"/>
                </xs:unique>
                <xs:unique name="parameterName">
                        <xs:selector xpath="./ns:parameter"/>
                        <xs:field xpath="ns:name"/>
                </xs:unique>
        </xs:element>
        <xs:element name="attribute">
                <xs:complexType>
                        <xs:complexContent>
                                <xs:extension base="attributeType"/>
                        </xs:complexContent>
                </xs:complexType>
        </xs:element>
        <xs:element name="parameter">
                <xs:complexType>
                        <xs:complexContent>
                                <xs:extension base="parameterType"/>
                        </xs:complexContent>
                </xs:complexType>
        </xs:element>
        <xs:complexType name="switchType">
                <xs:simpleContent>
                        <xs:extension base="enabled_disabledEnumerations">
                                <xs:attributeGroup ref="modelID:commonAttributes"/>
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>
        <xs:complexType name="tagType">
                <xs:sequence>
                        <xs:element name="dataType" type="ReferenceType">
                                <xs:annotation>
                                        <xs:documentation>identifies the datatype for the user-defined tag</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="semantics" type="modelID:String" minOccurs="0">
                                <xs:annotation>
                                        <xs:documentation>expands and describes the use of the datatype for the user-supplied
tag</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:any namespace="##other" minOccurs="0"/>
                </xs:sequence>
                <xs:attributeGroup ref="modelID:commonAttributes"/>
        </xs:complexType>
        <xs:complexType name="objectsType">
                <xs:sequence>
                        <xs:element ref="objectClass"/>
                        <xs:any namespace="##other" minOccurs="0"/>
                </xs:sequence>
                <xs:attributeGroup ref="modelID:commonAttributes"/>
        </xs:complexType>
        <xs:complexType name="interactionsType">
                <xs:sequence>
                        <xs:element ref="interactionClass"/>
```

```
                        <xs:any namespace="##other" minOccurs="0"/>
                    </xs:sequence>
                    <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:complexType>
                <xs:complexType name="attributeType">
                    <xs:sequence>
                        <xs:element name="name" type="modelID:IdentifierType"/>
                        <xs:element name="dataType" type="ReferenceType">
                            <xs:annotation>
                                <xs:documentation>identifies the datatype of the attribute</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:element name="updateType" type="updateType">
                            <xs:annotation>
                                <xs:documentation>records the policy for updating an instance of the class
attribute</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:element name="updateCondition" type="modelID:String" minOccurs="0">
                            <xs:annotation>
                                <xs:documentation>expands and explains the policies for updating an instance of the class
attribute</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:element name="ownership" type="ownershipType">
                            <xs:annotation>
                                <xs:documentation>indicates whether ownership of an instance of the class attribute can be
divested and/or acquired</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:element name="sharing" type="sharingType">
                            <xs:annotation>
                                <xs:documentation>identifies the capabilities of a federate or federation with respect to class
attribute publishing and subscribing</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:element name="dimensions" minOccurs="0">
                            <xs:annotation>
                                <xs:documentation>records the association of the class attribute with a set ofif a federate or
federation is using DDM services for this attribute</xs:documentation>
                            </xs:annotation>
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="dimension" type="ReferenceType" minOccurs="0"
maxOccurs="unbounded">
                                        <xs:annotation>
                                            <xs:documentation>identifies a dimension associated with this
attribute</xs:documentation>
                                        </xs:annotation>
                                    </xs:element>
                                    <xs:any namespace="##other" minOccurs="0"/>
                                </xs:sequence>
                                <xs:attributeGroup ref="modelID:commonAttributes"/>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="transportation" type="ReferenceType">
                            <xs:annotation>
                                <xs:documentation>specifies the type of transportation used with this
attribute</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:element name="order" type="orderType">
                            <xs:annotation>
                                <xs:documentation>specifies the order of delivery used with instances of this class
attribute</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:element name="semantics" type="modelID:String" minOccurs="0">
                            <xs:annotation>
                                <xs:documentation>lexicon entry for this attribute</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:any namespace="##other" minOccurs="0"/>
                    </xs:sequence>
                    <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:complexType>
                <xs:complexType name="parameterType">
                    <xs:sequence>
                        <xs:element name="name" type="modelID:IdentifierType"/>
```

```
                        <xs:element name="dataType" type="ReferenceType">
                                <xs:annotation>
                                        <xs:documentation>identifies the datatype of the parameter</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="semantics" type="modelID:String" minOccurs="0">
                                <xs:annotation>
                                        <xs:documentation>lexicon entry for the parameter</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:any namespace="##other" minOccurs="0"/>
                </xs:sequence>
                <xs:attributeGroup ref="modelID:commonAttributes"/>
        </xs:complexType>
        <xs:complexType name="objectClassType">
                <xs:sequence>
                        <xs:element name="name" type="modelID:IdentifierType"/>
                        <xs:element name="sharing" type="sharingType" default="Neither">
                                <xs:annotation>
                                        <xs:documentation>specifies publication and subscription capabilities of this object
class</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="semantics" type="modelID:String" minOccurs="0">
                                <xs:annotation>
                                        <xs:documentation>lexicon entry for this object class</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element ref="attribute" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element ref="objectClass" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:any namespace="##other" minOccurs="0"/>
                </xs:sequence>
                <xs:attributeGroup ref="modelID:commonAttributes"/>
        </xs:complexType>
        <xs:complexType name="interactionClassType">
                <xs:sequence>
                        <xs:element name="name" type="modelID:IdentifierType"/>
                        <xs:element name="sharing" type="sharingType">
                                <xs:annotation>
                                        <xs:documentation>specifies publication and subscription capabilities of this interaction
class</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="dimensions" minOccurs="0">
                                <xs:annotation>
                                        <xs:documentation>records the association of the interaction class with a set ofif a federate or
federation is using DDM services for this attribute</xs:documentation>
                                </xs:annotation>
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="dimension" type="ReferenceType" minOccurs="0"
maxOccurs="unbounded">
                                                        <xs:annotation>
                                                                <xs:documentation>identifies a dimension associated with this
interaction class</xs:documentation>
                                                        </xs:annotation>
                                                </xs:element>
                                                <xs:any namespace="##other" minOccurs="0"/>
                                        </xs:sequence>
                                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                                </xs:complexType>
                        </xs:element>
                        <xs:element name="transportation" type="ReferenceType">
                                <xs:annotation>
                                        <xs:documentation>specifies the type of transportation used with this interaction
class</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="order" type="orderType">
                                <xs:annotation>
                                        <xs:documentation>specifies the order of delivery used with instances of this interaction
class</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="semantics" type="modelID:String" minOccurs="0">
                                <xs:annotation>
                                        <xs:documentation>lexicon entry for this interaction class</xs:documentation>
                                </xs:annotation>
                        </xs:element>
```

```
                        <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element ref="interactionClass" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:any namespace="##other" minOccurs="0"/>
                    </xs:sequence>
                    <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:complexType>
                <xs:complexType name="dimensionsType">
                    <xs:sequence>
                        <xs:element name="dimension" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:complexContent>
                                    <xs:extension base="dimensionType"/>
                                </xs:complexContent>
                            </xs:complexType>
                        </xs:element>
                        <xs:any namespace="##other" minOccurs="0"/>
                    </xs:sequence>
                    <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:complexType>
                <xs:complexType name="dimensionType">
                    <xs:sequence>
                        <xs:element name="name" type="modelID:IdentifierType"/>
                        <xs:element name="dataType" type="ReferenceType">
                            <xs:annotation>
                                <xs:documentation>identifies the datatype for the federate view of the
dimension</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:element name="upperBound" minOccurs="0">
                            <xs:annotation>
                                <xs:documentation>specifies the upper bound for the dimension that meets the's requirement for
dimension subrange resolution</xs:documentation>
                            </xs:annotation>
                            <xs:complexType>
                                <xs:simpleContent>
                                    <xs:extension base="xs:positiveInteger">
                                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                                    </xs:extension>
                                </xs:simpleContent>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="normalization" type="modelID:NonEmptyString">
                            <xs:annotation>
                                <xs:documentation>specifies the map from a subscription/update region'scoordinates to
nonnegative integer subranges in the range [0, dimension upper bound).</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:element name="value" type="dimensionValuePattern">
                            <xs:annotation>
                                <xs:documentation>specifies a default range for the dimension that the RTI is to use in overlap
calculations if the dimension is an available dimension of an attribute or interaction and has been left unspecified when a federate creates a region that is
subsequently used with the attribute or interaction</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:any namespace="##other" minOccurs="0"/>
                    </xs:sequence>
                    <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:complexType>
                <xs:complexType name="timeType">
                    <xs:sequence>
                        <xs:element name="timeStamp" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="dataType" type="ReferenceType">
                                        <xs:annotation>
                                            <xs:documentation>identifies the timestamp
datatype</xs:documentation>
                                        </xs:annotation>
                                    </xs:element>
                                    <xs:element name="semantics" type="modelID:String" minOccurs="0">
                                        <xs:annotation>
                                            <xs:documentation>expands and describes the use of the
datatype for timestamp</xs:documentation>
                                        </xs:annotation>
                                    </xs:element>
                                    <xs:any namespace="##other" minOccurs="0"/>
                                </xs:sequence>
                                <xs:attributeGroup ref="modelID:commonAttributes"/>
                            </xs:complexType>
```

```
                        </xs:element>
                        <xs:element name="lookahead" minOccurs="0">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="dataType" type="ReferenceType">
                                                        <xs:annotation>
                                                                <xs:documentation>identifies the lookahead
datatype</xs:documentation>
                                                        </xs:annotation>
                                                </xs:element>
                                                <xs:element name="semantics" type="modelID:String" minOccurs="0">
                                                        <xs:annotation>
                                                                <xs:documentation>expands and describes the use of the
datatype for lookahead</xs:documentation>
                                                        </xs:annotation>
                                                </xs:element>
                                                <xs:any namespace="##other" minOccurs="0"/>
                                        </xs:sequence>
                                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                                </xs:complexType>
                        </xs:element>
                        <xs:any namespace="##other" minOccurs="0"/>
                </xs:sequence>
                <xs:attributeGroup ref="modelID:commonAttributes"/>
        </xs:complexType>
        <xs:complexType name="tagsType">
                <xs:sequence>
                        <xs:element name="updateReflectTag" type="tagType" minOccurs="0"/>
                        <xs:element name="sendReceiveTag" type="tagType" minOccurs="0"/>
                        <xs:element name="deleteRemoveTag" type="tagType" minOccurs="0"/>
                        <xs:element name="divestitureRequestTag" type="tagType" minOccurs="0"/>
                        <xs:element name="divestitureCompletionTag" type="tagType" minOccurs="0"/>
                        <xs:element name="acquisitionRequestTag" type="tagType" minOccurs="0"/>
                        <xs:element name="requestUpdateTag" type="tagType" minOccurs="0"/>
                        <xs:any namespace="##other" minOccurs="0"/>
                </xs:sequence>
                <xs:attributeGroup ref="modelID:commonAttributes"/>
        </xs:complexType>
        <xs:complexType name="synchronizationPointType">
                <xs:sequence>
                        <xs:element name="label" type="modelID:IdentifierType"/>
                        <xs:element name="dataType" type="ReferenceType" minOccurs="0">
                                <xs:annotation>
                                        <xs:documentation>identifies the datatype of the user-supplied tag for those synchronizationthat
the federate or federation designate as providing a user-supplied tag</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="capability" type="capabilityType">
                                <xs:annotation>
                                        <xs:documentation>indicates the level of interaction that a federate is capable of
honoring</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="semantics" type="modelID:String" minOccurs="0">
                                <xs:annotation>
                                        <xs:documentation>expands and describes the use of the synchronization
point</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:any namespace="##other" minOccurs="0"/>
                </xs:sequence>
                <xs:attributeGroup ref="modelID:commonAttributes"/>
        </xs:complexType>
        <xs:complexType name="synchronizationsType">
                <xs:sequence>
                        <xs:element name="synchronizationPoint" minOccurs="0" maxOccurs="unbounded">
                                <xs:complexType>
                                        <xs:complexContent>
                                                <xs:extension base="synchronizationPointType"/>
                                        </xs:complexContent>
                                </xs:complexType>
                        </xs:element>
                        <xs:any namespace="##other" minOccurs="0"/>
                </xs:sequence>
                <xs:attributeGroup ref="modelID:commonAttributes"/>
        </xs:complexType>
        <xs:complexType name="transportationsType">
                <xs:sequence>
                        <xs:element name="transportation" minOccurs="0" maxOccurs="unbounded">
```

```
                                        <xs:complexType>
                                                <xs:complexContent>
                                                        <xs:extension base="transportationType"/>
                                                </xs:complexContent>
                                        </xs:complexType>
                                </xs:element>
                                <xs:any namespace="##other" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:complexType>
                <xs:complexType name="transportationType">
                        <xs:sequence>
                                <xs:element name="name" type="modelID:IdentifierType"/>
                                <xs:element name="semantics" type="modelID:String" minOccurs="0">
                                        <xs:annotation>
                                                <xs:documentation>describes the transportation type</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:any namespace="##other" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:complexType>
                <xs:complexType name="switchesType">
                        <xs:sequence>
                                <xs:element name="autoProvide" type="switchType" minOccurs="0"/>
                                <xs:element name="conveyRegionDesignatorSets" type="switchType" minOccurs="0"/>
                                <xs:element name="attributeScopeAdvisory" type="switchType" minOccurs="0"/>
                                <xs:element name="attributeRelevanceAdvisory" type="switchType" minOccurs="0"/>
                                <xs:element name="objectClassRelevanceAdvisory" type="switchType" minOccurs="0"/>
                                <xs:element name="interactionRelevanceAdvisory" type="switchType" minOccurs="0"/>
                                <xs:element name="serviceReporting" type="switchType" minOccurs="0"/>
                                <xs:element name="automaticResign" type="switchType" minOccurs="0"/>
                                <xs:any namespace="##other" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:complexType>
                <xs:complexType name="basicDataType">
                        <xs:sequence>
                                <xs:element name="name" type="modelID:IdentifierType"/>
                                <xs:element name="size" type="Size">
                                        <xs:annotation>
                                                <xs:documentation>defines the size of the data representation in terms of the number of bits
contained in the data representation</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="interpretation" type="modelID:String">
                                        <xs:annotation>
                                                <xs:documentation>describes how the data representation should be
interpreted</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="endian" type="endianType">
                                        <xs:annotation>
                                                <xs:documentation>describe how multiple bytes within the representation are
arranged</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="encoding" type="modelID:String">
                                        <xs:annotation>
                                                <xs:documentation>describes, in detail, the encoding of the data representation (e.g., the bit) as
delivered to and received from the RTI</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:any namespace="##other" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:complexType>
                <xs:complexType name="simpleDataType">
                        <xs:sequence>
                                <xs:element name="name" type="modelID:IdentifierType"/>
                                <xs:element name="representation" type="ReferenceType">
                                        <xs:annotation>
                                                <xs:documentation>identifies the basic data representation of this datatype</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="units" type="modelID:NonEmptyString" default="NA" minOccurs="0">
                                        <xs:annotation>
                                                <xs:documentation>identifies the units of measure (e.g., m, km, kg) for the datatype whenever
such units exist. Any units entered in this column also specify the units of Resolution and Accuracy.</xs:documentation>
```

```
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="resolution" type="modelID:NonEmptyString" minOccurs="0">
                                        <xs:annotation>
                                                <xs:documentation>describes the precision of measure for the datatype</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="accuracy" type="modelID:NonEmptyString" minOccurs="0">
                                        <xs:annotation>
                                                <xs:documentation>describes maximum deviation of the value from its intended value in theor
federation</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
                                <xs:any namespace="##other" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:complexType>
                <xs:complexType name="fixedRecordDataType">
                        <xs:sequence>
                                <xs:element name="name" type="modelID:IdentifierType"/>
                                <xs:element name="encoding" type="fixedRecordEncodingType">
                                        <xs:annotation>
                                                <xs:documentation>describes the encoding of the fixed record datatype (i.e., the organization of
fields) as delivered to and received from the RTI</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
                                <xs:element name="field" minOccurs="0" maxOccurs="unbounded">
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element name="name" type="modelID:IdentifierType"/>
                                                        <xs:element name="dataType" type="ReferenceType"/>
                                                        <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
                                                        <xs:any namespace="##other" minOccurs="0"/>
                                                </xs:sequence>
                                                <xs:attributeGroup ref="modelID:commonAttributes"/>
                                        </xs:complexType>
                                </xs:element>
                                <xs:any namespace="##other" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:complexType>
                <xs:complexType name="enumeratedDataType">
                        <xs:sequence>
                                <xs:element name="name" type="modelID:IdentifierType"/>
                                <xs:element name="representation" type="ReferenceType">
                                        <xs:annotation>
                                                <xs:documentation>identifies the basic data representation that forms the basis of
this</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
                                <xs:element name="enumerator" minOccurs="0" maxOccurs="unbounded">
                                        <xs:annotation>
                                                <xs:documentation>defines the enumerators associated with this datatype</xs:documentation>
                                        </xs:annotation>
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element name="name" type="modelID:IdentifierType"/>
                                                        <xs:element name="value" type="modelID:String" maxOccurs="unbounded">
                                                                <xs:annotation>
                                                                        <xs:documentation>provides values that correspond to each
enumerator</xs:documentation>
                                                                </xs:annotation>
                                                        </xs:element>
                                                        <xs:any namespace="##other" minOccurs="0"/>
                                                </xs:sequence>
                                                <xs:attributeGroup ref="modelID:commonAttributes"/>
                                        </xs:complexType>
                                </xs:element>
                                <xs:any namespace="##other" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:complexType>
                <xs:complexType name="arrayDataType">
                        <xs:sequence>
                                <xs:element name="name" type="modelID:IdentifierType"/>
                                <xs:element name="dataType" type="ReferenceType">
```

```
                                        <xs:annotation>
                                                <xs:documentation>identifies the datatype of an element of this array</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="cardinality" type="cardinalityType">
                                        <xs:annotation>
                                                <xs:documentation>contains the number of elements that are contained in the
array</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="encoding" type="arrayDatatypeEncodingType">
                                        <xs:annotation>
                                                <xs:documentation>describe, in detail, the encoding of the array datatype (e.g., the
sequenceelements and the order of elements in multi-dimensional arrays) as delivered to and received from the</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
                                <xs:any namespace="##other" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:complexType>
                <xs:complexType name="variantRecordDataType">
                        <xs:sequence>
                                <xs:element name="name" type="modelID:IdentifierType"/>
                                <xs:element name="discriminant" type="modelID:IdentifierType"/>
                                <xs:element name="dataType" type="ReferenceType">
                                        <xs:annotation>
                                                <xs:documentation>identifies the datatype of the discriminant</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="alternative" minOccurs="0" maxOccurs="unbounded">
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element name="enumerator" type="modelID:NonEmptyString">
                                                                <xs:annotation>
                                                                        <xs:documentation>enumerators or enumerator ranges that
determines the alternative</xs:documentation>
                                                                </xs:annotation>
                                                        </xs:element>
                                                        <xs:element name="name" type="modelID:IdentifierType" minOccurs="0"/>
                                                        <xs:element name="dataType" type="ReferenceType" minOccurs="0">
                                                                <xs:annotation>
                                                                        <xs:documentation>identify the datatype of the
field</xs:documentation>
                                                                </xs:annotation>
                                                        </xs:element>
                                                        <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
                                                        <xs:any namespace="##other" minOccurs="0"/>
                                                </xs:sequence>
                                                <xs:attributeGroup ref="modelID:commonAttributes"/>
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="encoding" type="variantRecordEncodingType">
                                        <xs:annotation>
                                                <xs:documentation>describes the encoding of the variant record datatype (e.g., the location of
the discriminant) as delivered to and received from the RTI</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
                                <xs:any namespace="##other" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:complexType>
                <xs:complexType name="notesType">
                        <xs:sequence>
                                <xs:element name="note" minOccurs="0" maxOccurs="unbounded">
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element name="label" type="xs:ID"/>
                                                        <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
                                                        <xs:any namespace="##other" minOccurs="0"/>
                                                </xs:sequence>
                                                <xs:attributeGroup ref="modelID:commonAttributes"/>
                                        </xs:complexType>
                                </xs:element>
                                <xs:any namespace="##other" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:complexType>
```

```
<xs:complexType name="dataTypesType">
	<xs:sequence>
		<xs:element name="basicDataRepresentations" type="basicDataRepresentationsType">
			<xs:annotation>
				<xs:documentation>Basic data representation is the underpinning of all OMT datatypes. Although it is not used as a datatype, it forms the basis of the datatypes.</xs:documentation>
			</xs:annotation>
		</xs:element>
		<xs:element name="simpleDataTypes" type="simpleDataTypesType">
			<xs:annotation>
				<xs:documentation>The simple datatypes describes simple, scalar data items.</xs:documentation>
			</xs:annotation>
		</xs:element>
		<xs:element name="enumeratedDataTypes" type="enumeratedDataTypesType">
			<xs:annotation>
				<xs:documentation>The enumerated datatypes describes data elements that can take on a finite discrete set ofvalues.</xs:documentation>
			</xs:annotation>
		</xs:element>
		<xs:element name="arrayDataTypes">
			<xs:annotation>
				<xs:documentation>The array datatypes describes indexed homogenous collections of datatypes; also known as arrays or sequences.</xs:documentation>
			</xs:annotation>
			<xs:complexType>
				<xs:complexContent>
					<xs:extension base="arrayDataTypesType"/>
				</xs:complexContent>
			</xs:complexType>
		</xs:element>
		<xs:element name="fixedRecordDataTypes" type="fixedRecordDataTypesType">
			<xs:annotation>
				<xs:documentation>The fixed datatypes describes heterogeneous collections of types; also known as records or structures.</xs:documentation>
			</xs:annotation>
		</xs:element>
		<xs:element name="variantRecordDataTypes" type="variantRecordDataTypesType">
			<xs:annotation>
				<xs:documentation>The variant record datatypes describes discriminated unions of types; also known as variant or choice records.</xs:documentation>
			</xs:annotation>
		</xs:element>
		<xs:any namespace="##other" minOccurs="0"/>
	</xs:sequence>
	<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="basicDataRepresentationsType">
	<xs:sequence>
		<xs:element name="basicData" minOccurs="0" maxOccurs="unbounded">
			<xs:complexType>
				<xs:complexContent>
					<xs:extension base="basicDataType"/>
				</xs:complexContent>
			</xs:complexType>
		</xs:element>
		<xs:any namespace="##other" minOccurs="0"/>
	</xs:sequence>
	<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="simpleDataTypesType">
	<xs:sequence>
		<xs:element name="simpleData" minOccurs="0" maxOccurs="unbounded">
			<xs:complexType>
				<xs:complexContent>
					<xs:extension base="simpleDataType"/>
				</xs:complexContent>
			</xs:complexType>
		</xs:element>
		<xs:any namespace="##other" minOccurs="0"/>
	</xs:sequence>
	<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="enumeratedDataTypesType">
	<xs:sequence>
		<xs:element name="enumeratedData" minOccurs="0" maxOccurs="unbounded">
			<xs:complexType>
				<xs:complexContent>
					<xs:extension base="enumeratedDataType"/>
```

```
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:any namespace="##other" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="modelID:commonAttributes"/>
                        </xs:complexType>
                        <xs:complexType name="arrayDataTypesType">
                            <xs:sequence>
                                <xs:element name="arrayData" minOccurs="0" maxOccurs="unbounded">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:extension base="arrayDataType"/>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:any namespace="##other" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="modelID:commonAttributes"/>
                        </xs:complexType>
                        <xs:complexType name="fixedRecordDataTypesType">
                            <xs:sequence>
                                <xs:element name="fixedRecordData" minOccurs="0" maxOccurs="unbounded">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:extension base="fixedRecordDataType"/>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:any namespace="##other" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="modelID:commonAttributes"/>
                        </xs:complexType>
                        <xs:complexType name="variantRecordDataTypesType">
                            <xs:sequence>
                                <xs:element name="variantRecordData" minOccurs="0" maxOccurs="unbounded">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:extension base="variantRecordDataType"/>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:any namespace="##other" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="modelID:commonAttributes"/>
                        </xs:complexType>
                        <xs:complexType name="objectModelType">
                            <xs:sequence>
                                <xs:element name="modelIdentification" type="modelID:modelIdentificationType">
                                    <xs:annotation>
                                        <xs:documentation>documents certain key identifying information within the object model
description</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                                <xs:element name="conformance" type="conformanceType" minOccurs="0">
                                    <xs:annotation>
                                        <xs:documentation>conformance statement</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                                <xs:element name="objects" type="objectsType">
                                    <xs:annotation>
                                        <xs:documentation>specifies classes of objects and their hierarchical relationships
</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                                <xs:element name="interactions" type="interactionsType">
                                    <xs:annotation>
                                        <xs:documentation>specifies classes of interactions and their hierarchical relationships
</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                                <xs:element name="dimensions" type="dimensionsType">
                                    <xs:annotation>
                                        <xs:documentation>specifies dimensions associated with attribute types and interaction
classes</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                                <xs:element name="time" type="timeType">
                                    <xs:annotation>
```

```
                                        <xs:documentation>specifies timestamp and lookahead datatypes</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                                <xs:element name="tags" type="tagsType">
                                    <xs:annotation>
                                        <xs:documentation>specifies the datatype of user-defined tags</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                                <xs:element name="synchronizations" type="synchronizationsType">
                                    <xs:annotation>
                                        <xs:documentation>specifies federate and federation capabilities for synchronization-
points</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                                <xs:element name="transportations" type="transportationsType">
                                    <xs:annotation>
                                        <xs:documentation>documents transportation type support and agreements</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                                <xs:element name="switches" type="switchesType">
                                    <xs:annotation>
                                        <xs:documentation>specification of the initial setting of RTI switches</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                                <xs:element name="updateRates" type="updateRatesType"/>
                                <xs:element name="dataTypes" type="dataTypesType">
                                    <xs:annotation>
                                        <xs:documentation>specifies all referenced datatypes</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                                <xs:element name="notes" type="notesType">
                                    <xs:annotation>
                                        <xs:documentation>specifies all referenced notes</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                                <xs:any namespace="##other" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="modelID:commonAttributes"/>
                        </xs:complexType>
                        <xs:complexType name="Size">
                            <xs:simpleContent>
                                <xs:extension base="xs:integer">
                                    <xs:attributeGroup ref="modelID:commonAttributes"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                        <xs:complexType name="RateType">
                            <xs:simpleContent>
                                <xs:extension base="xs:float">
                                    <xs:attributeGroup ref="modelID:commonAttributes"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                        <xs:complexType name="ReferenceType">
                            <xs:simpleContent>
                                <xs:extension base="xs:NCName">
                                    <xs:attributeGroup ref="modelID:commonAttributes"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                        <xs:complexType name="sharingType">
                            <xs:simpleContent>
                                <xs:extension base="sharingEnumerations">
                                    <xs:attributeGroup ref="modelID:commonAttributes"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                        <xs:complexType name="orderType">
                            <xs:simpleContent>
                                <xs:extension base="orderEnumerations">
                                    <xs:attributeGroup ref="modelID:commonAttributes"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                        <xs:complexType name="updateType">
                            <xs:simpleContent>
                                <xs:extension base="updateEnumerations">
                                    <xs:attributeGroup ref="modelID:commonAttributes"/>
                                </xs:extension>
```

```
		</xs:simpleContent>
</xs:complexType>
<xs:complexType name="endianType">
		<xs:simpleContent>
			<xs:extension base="endianEnumerations">
				<xs:attributeGroup ref="modelID:commonAttributes"/>
			</xs:extension>
		</xs:simpleContent>
</xs:complexType>
<xs:complexType name="ownershipType">
		<xs:simpleContent>
			<xs:extension base="ownershipEnumerations">
				<xs:attributeGroup ref="modelID:commonAttributes"/>
			</xs:extension>
		</xs:simpleContent>
</xs:complexType>
<xs:complexType name="capabilityType">
		<xs:simpleContent>
			<xs:extension base="capabilityEnumerations">
				<xs:attributeGroup ref="modelID:commonAttributes"/>
			</xs:extension>
		</xs:simpleContent>
</xs:complexType>
<xs:simpleType name="sharingEnumerations">
		<xs:restriction base="xs:string">
			<xs:enumeration value="Publish"/>
			<xs:enumeration value="Subscribe"/>
			<xs:enumeration value="PublishSubscribe"/>
			<xs:enumeration value="Neither"/>
		</xs:restriction>
</xs:simpleType>
<xs:simpleType name="orderEnumerations">
		<xs:annotation>
			<xs:documentation> </xs:documentation>
		</xs:annotation>
		<xs:restriction base="xs:string">
			<xs:enumeration value="Receive"/>
			<xs:enumeration value="TimeStamp"/>
		</xs:restriction>
</xs:simpleType>
<xs:simpleType name="endianEnumerations">
		<xs:restriction base="xs:string">
			<xs:enumeration value="Big"/>
			<xs:enumeration value="Little"/>
		</xs:restriction>
</xs:simpleType>
<xs:simpleType name="capabilityEnumerations">
		<xs:restriction base="xs:string">
			<xs:enumeration value="Register"/>
			<xs:enumeration value="Achieve"/>
			<xs:enumeration value="RegisterAchieve"/>
			<xs:enumeration value="NoSynch"/>
			<xs:enumeration value="NA"/>
		</xs:restriction>
</xs:simpleType>
<xs:simpleType name="updateEnumerations">
		<xs:restriction base="xs:string">
			<xs:enumeration value="Static"/>
			<xs:enumeration value="Periodic"/>
			<xs:enumeration value="Conditional"/>
			<xs:enumeration value="NA"/>
		</xs:restriction>
</xs:simpleType>
<xs:simpleType name="ownershipEnumerations">
		<xs:annotation>
			<xs:documentation> </xs:documentation>
		</xs:annotation>
		<xs:restriction base="xs:string">
			<xs:enumeration value="Divest"/>
			<xs:enumeration value="Acquire"/>
			<xs:enumeration value="DivestAcquire"/>
			<xs:enumeration value="NoTransfer"/>
		</xs:restriction>
</xs:simpleType>
<xs:simpleType name="enabled_disabledEnumerations">
		<xs:restriction base="xs:string">
			<xs:enumeration value="Enabled"/>
			<xs:enumeration value="Disabled"/>
		</xs:restriction>
```

```
		</xs:simpleType>
		<xs:complexType name="updateRateType">
			<xs:sequence>
				<xs:element name="name" type="modelID:IdentifierType"/>
				<xs:element name="rate" type="RateType"/>
				<xs:element name="semantics" type="modelID:String" minOccurs="0"/>
				<xs:any namespace="##other" minOccurs="0"/>
			</xs:sequence>
			<xs:attributeGroup ref="modelID:commonAttributes"/>
		</xs:complexType>
		<xs:complexType name="updateRatesType">
			<xs:sequence>
				<xs:element name="updateRate" type="updateRateType" minOccurs="0" maxOccurs="unbounded"/>
				<xs:any namespace="##other" minOccurs="0"/>
			</xs:sequence>
			<xs:attributeGroup ref="modelID:commonAttributes"/>
		</xs:complexType>
		<xs:complexType name="conformanceType">
			<xs:sequence>
				<xs:element name="implementedServices">
					<xs:annotation>
						<xs:documentation>RTI services implemented/used in the federation or by a
federate</xs:documentation>
					</xs:annotation>
					<xs:complexType>
						<xs:all>
							<!-- Federation Management Services</xs:documentation -->
							<xs:element name="createFederationExecution" minOccurs="0"/>
							<xs:element name="destroyFederationExecution" minOccurs="0"/>
							<xs:element name="joinFederatioExecution" minOccurs="0"/>
							<xs:element name="resignFederationExecution" minOccurs="0"/>
							<xs:element name="registerFederationSynchronizationPoint" minOccurs="0"/>
							<xs:element name="confirmSynchronizationPointRegistration" minOccurs="0"/>
							<xs:element name="announceSynchronizationPoint" minOccurs="0"/>
							<xs:element name="synchronizationPointAchieved" minOccurs="0"/>
							<xs:element name="federationSynchronized" minOccurs="0"/>
							<xs:element name="requestFederationSave" minOccurs="0"/>
							<xs:element name="initiateFederateSave" minOccurs="0"/>
							<xs:element name="federateSaveBegun" minOccurs="0"/>
							<xs:element name="federateSaveComplete" minOccurs="0"/>
							<xs:element name="federationSaved" minOccurs="0"/>
							<xs:element name="queryFederationSaveStatus" minOccurs="0"/>
							<xs:element name="federationSaveStatusResponse" minOccurs="0"/>
							<xs:element name="requestFederationRestore" minOccurs="0"/>
							<xs:element name="confirmFederationRestorationRequest" minOccurs="0"/>
							<xs:element name="federationRestoreBegun" minOccurs="0"/>
							<xs:element name="initiateFederateRestore" minOccurs="0"/>
							<xs:element name="federateRestoreComplete" minOccurs="0"/>
							<xs:element name="federationRestored" minOccurs="0"/>
							<xs:element name="queryFederationRestoreStatus" minOccurs="0"/>
							<xs:element name="federationRestoreStatusResponse" minOccurs="0"/>
							<xs:element name="connect" minOccurs="0"/>
							<xs:element name="disconnect" minOccurs="0"/>
							<xs:element name="connectionLost" minOccurs="0"/>
							<!-- Declaration Management -->
							<xs:element name="publishObjectClassAttributes" minOccurs="0"/>
							<xs:element name="unpublishObjectClassAttributes" minOccurs="0"/>
							<xs:element name="publishInteractionClass" minOccurs="0"/>
							<xs:element name="unpublishInteractionClass" minOccurs="0"/>
							<xs:element name="subscribeObjectClassAttributes" minOccurs="0"/>
							<xs:element name="unsubscribeObjectClassAttributes" minOccurs="0"/>
							<xs:element name="subscribeInteractionClass" minOccurs="0"/>
							<xs:element name="unsubscribeInteractionClass" minOccurs="0"/>
							<xs:element name="startRegistrationForObjectClass" minOccurs="0"/>
							<xs:element name="stopRegistrationForObjectClass" minOccurs="0"/>
							<xs:element name="turnInteractionsOn" minOccurs="0"/>
							<xs:element name="turnInteractionsOff" minOccurs="0"/>
							<!-- Object Management -->
							<xs:element name="reserveObjectInstanceName" minOccurs="0"/>
							<xs:element name="objectInstanceNameReserved" minOccurs="0"/>
							<xs:element name="registerObjectInstance" minOccurs="0"/>
							<xs:element name="discoverObjectInstance" minOccurs="0"/>
							<xs:element name="updateAttributeValues" minOccurs="0"/>
							<xs:element name="reflectAttributeValues" minOccurs="0"/>
							<xs:element name="sendInteraction" minOccurs="0"/>
							<xs:element name="receiveInteraction" minOccurs="0"/>
							<xs:element name="deleteObjectInstance" minOccurs="0"/>
							<xs:element name="removeObjectInstance" minOccurs="0"/>
							<xs:element name="localDeleteObjectInstance" minOccurs="0"/>
```

```
                                        <xs:element name="changeAttributeTransportationType" minOccurs="0"/>
                                        <xs:element name="changeInteractionTransportationType" minOccurs="0"/>
                                        <xs:element name="attributesInScope" minOccurs="0"/>
                                        <xs:element name="attributesOutOfScope" minOccurs="0"/>
                                        <xs:element name="requestAttributeValueUpdate" minOccurs="0"/>
                                        <xs:element name="provideAttributeValueUpdate" minOccurs="0"/>
                                        <xs:element name="turnUpdatesOnForObjectInstance" minOccurs="0"/>
                                        <xs:element name="turnUpdatesOffForObjectInstance" minOccurs="0"/>
                                        <!-- Ownership Management -->
                                        <xs:element name="unconditionalAttributeOwnershipDivestiture" minOccurs="0"/>
                                        <xs:element name="negotiatedAttributeOwnershipDivestiture" minOccurs="0"/>
                                        <xs:element name="requestAttributeOwnershipAssumption" minOccurs="0"/>
                                        <xs:element name="requestDivestitureConfirmation" minOccurs="0"/>
                                        <xs:element name="confirmDivestiture" minOccurs="0"/>
                                        <xs:element name="attributeOwnershipAcquisitionNotification" minOccurs="0"/>
                                        <xs:element name="attributeOwnershipAcquisition" minOccurs="0"/>
                                        <xs:element name="attributeOwnershipAcquisitionIfAvailable" minOccurs="0"/>
                                        <xs:element name="attributeOwnershipUnavailable" minOccurs="0"/>
                                        <xs:element name="requestAttributeOwnershipRelease" minOccurs="0"/>
                                        <xs:element name="attributeOwnershipReleaseDenied" minOccurs="0"/>
                                        <xs:element name="attributeOwnershipDivestitureIfWanted" minOccurs="0"/>
                                        <xs:element name="cancelNegotiatedAttributeOwnershipDivestiture"
minOccurs="0"/>
                                        <xs:element name="cancelAttributeOwnershipAcquisition" minOccurs="0"/>
                                        <xs:element name="confirmAttributeOwnershipAcquisitionCancellation"
minOccurs="0"/>
                                        <xs:element name="queryAttributeOwnership" minOccurs="0"/>
                                        <xs:element name="informAttributeOwnership" minOccurs="0"/>
                                        <xs:element name="isAttributeOwnedByFederate" minOccurs="0"/>
                                        <!-- Time Management -->
                                        <xs:element name="enableTimeRegulation" minOccurs="0"/>
                                        <xs:element name="timeRegulationEnabled" minOccurs="0"/>
                                        <xs:element name="disableTimeRegulation" minOccurs="0"/>
                                        <xs:element name="enableTimeConstrained" minOccurs="0"/>
                                        <xs:element name="timeConstrainedEnabled" minOccurs="0"/>
                                        <xs:element name="disableTimeConstrained" minOccurs="0"/>
                                        <xs:element name="timeAdvanceRequest" minOccurs="0"/>
                                        <xs:element name="timeAdvanceRequestAvailable" minOccurs="0"/>
                                        <xs:element name="nextMessageRequest" minOccurs="0"/>
                                        <xs:element name="nextMessageRequestAvailable" minOccurs="0"/>
                                        <xs:element name="flushQueueRequest" minOccurs="0"/>
                                        <xs:element name="timeAdvanceGrant" minOccurs="0"/>
                                        <xs:element name="enableAsynchronousDelivery" minOccurs="0"/>
                                        <xs:element name="disableAsynchronousDelivery" minOccurs="0"/>
                                        <xs:element name="queryGALT" minOccurs="0"/>
                                        <xs:element name="queryLogicalTime" minOccurs="0"/>
                                        <xs:element name="queryLITS" minOccurs="0"/>
                                        <xs:element name="modifyLookahead" minOccurs="0"/>
                                        <xs:element name="queryLookahead" minOccurs="0"/>
                                        <xs:element name="retract" minOccurs="0"/>
                                        <xs:element name="requestRetraction" minOccurs="0"/>
                                        <xs:element name="changeAttributeOrderType" minOccurs="0"/>
                                        <xs:element name="changeInteractionOrderType" minOccurs="0"/>
                                        <!-- Data Distribution Management -->
                                        <xs:element name="createRegion" minOccurs="0"/>
                                        <xs:element name="commitRegionModifications" minOccurs="0"/>
                                        <xs:element name="deleteRegion" minOccurs="0"/>
                                        <xs:element name="registerObjectInstanceWithRegions" minOccurs="0"/>
                                        <xs:element name="associateRegionForUpdates" minOccurs="0"/>
                                        <xs:element name="unassociateRegionForUpdates" minOccurs="0"/>
                                        <xs:element name="subscribeObjectClassAttributesWithRegions" minOccurs="0"/>
                                        <xs:element name="unsubscribeObjectClassAttributesWithRegions" minOccurs="0"/>
                                        <xs:element name="subscribeInteractionClassWithRegions" minOccurs="0"/>
                                        <xs:element name="unsubscribeInteractionClassWithRegions" minOccurs="0"/>
                                        <xs:element name="sendInteractionWithRegions" minOccurs="0"/>
                                        <xs:element name="requestAttributeValueUpdateWithRegions" minOccurs="0"/>
                                        <!-- Support Services -->
                                        <xs:element name="getObjectClassHandle" minOccurs="0"/>
                                        <xs:element name="getObjectClassName" minOccurs="0"/>
                                        <xs:element name="getAttributeHandle" minOccurs="0"/>
                                        <xs:element name="getAttributeName" minOccurs="0"/>
                                        <xs:element name="getInteractionClassHandle" minOccurs="0"/>
                                        <xs:element name="getInteractionClassName" minOccurs="0"/>
                                        <xs:element name="getParameterHandle" minOccurs="0"/>
                                        <xs:element name="getParameterName" minOccurs="0"/>
                                        <xs:element name="getObjectInstanceHandle" minOccurs="0"/>
                                        <xs:element name="getObjectInstanceName" minOccurs="0"/>
                                        <xs:element name="getDimensionHandle" minOccurs="0"/>
                                        <xs:element name="getDimensionName" minOccurs="0"/>
```

```
                                        <xs:element name="getDimensionUpperBound" minOccurs="0"/>
                                        <xs:element name="getAvailableDimensionsForClassAttribute" minOccurs="0"/>
                                        <xs:element name="getKnownObjectClassHandle" minOccurs="0"/>
                                        <xs:element name="getAvailableDimensionsForInteractionClass" minOccurs="0"/>
                                        <xs:element name="getTransportationType" minOccurs="0"/>
                                        <xs:element name="getTransportationName" minOccurs="0"/>
                                        <xs:element name="getOrderType" minOccurs="0"/>
                                        <xs:element name="getOrderName" minOccurs="0"/>
                                        <xs:element name="enableObjectClassRelevanceAdvisorySwitch" minOccurs="0"/>
                                        <xs:element name="disableObjectClassRelevanceAdvisorySwitch" minOccurs="0"/>
                                        <xs:element name="enableAttributeRelevanceAdvisorySwitch" minOccurs="0"/>
                                        <xs:element name="disableAttributeRelevanceAdvisorySwitch" minOccurs="0"/>
                                        <xs:element name="enableAttributeScopeAdvisorySwitch" minOccurs="0"/>
                                        <xs:element name="disableAttributeScopeAdvisorySwitch" minOccurs="0"/>
                                        <xs:element name="enableInteractionRelevanceAdvisorySwitch" minOccurs="0"/>
                                        <xs:element name="disableInteractionRelevanceAdvisorySwitch" minOccurs="0"/>
                                        <xs:element name="getDimensionHandleSet" minOccurs="0"/>
                                        <xs:element name="getRangeBounds" minOccurs="0"/>
                                        <xs:element name="setRangeBounds" minOccurs="0"/>
                                        <xs:element name="normalizeFederateHandle" minOccurs="0"/>
                                        <xs:element name="normalizeServiceGroup" minOccurs="0"/>
                                        <xs:element name="evokeCallback" minOccurs="0"/>
                                        <xs:element name="evokeMultipleCallback" minOccurs="0"/>
                                        <xs:element name="enableCallbacks" minOccurs="0"/>
                                        <xs:element name="disableCallbacks" minOccurs="0"/>
                                        <xs:element name="getAutomaticResignDirective" minOccurs="0"/>
                                        <xs:element name="setAutomaticResignDirective" minOccurs="0"/>
                                        <xs:element name="getFederateHandle" minOccurs="0"/>
                                        <xs:element name="getFederateName" minOccurs="0"/>
                                        <xs:element name="getUpdateRateValueForAttribute" minOccurs="0"/>
                                </xs:all>
                                <xs:attributeGroup ref="modelID:commonAttributes"/>
                        </xs:complexType>
                </xs:element>
                <xs:any namespace="##other" minOccurs="0"/>
        </xs:sequence>
        <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:simpleType name="dimensionValuePattern">
        <xs:restriction base="xs:string">
                <!--xs:pattern value="(\d)+|[(\d)+..(\d)+)|Excluded"/-->
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="cardinalityPattern">
        <xs:restriction base="xs:string">
                <xs:pattern value="(Dynamic|(\d)+|(\[(\d)+..(\d)+\]))(,(Dynamic|(\d)+|(\[(\d)+..(\d)+\])))*"/>
        </xs:restriction>
</xs:simpleType>
<xs:complexType name="cardinalityType">
        <xs:simpleContent>
                <xs:extension base="cardinalityPattern">
                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:extension>
        </xs:simpleContent>
</xs:complexType>
<xs:simpleType name="arrayDatatypeEncodingEnum">
        <xs:restriction base="xs:string">
                <xs:pattern value="HLAfixedArray"/>
                <xs:pattern value="HLAvariableArray"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="arrayDatatypeEncodingUnion">
        <xs:union memberTypes="arrayDatatypeEncodingEnum modelID:nonEmptyString"/>
</xs:simpleType>
<xs:complexType name="arrayDatatypeEncodingType">
        <xs:simpleContent>
                <xs:extension base="arrayDatatypeEncodingUnion">
                        <xs:attributeGroup ref="modelID:commonAttributes"/>
                </xs:extension>
        </xs:simpleContent>
</xs:complexType>
<xs:simpleType name="fixedRecordEncodingEnumeration">
        <xs:restriction base="xs:string">
                <xs:enumeration value="HLAfixedRecord"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="fixedRecordEncodingUnion">
        <xs:union memberTypes="fixedRecordEncodingEnumeration modelID:nonEmptyString"/>
</xs:simpleType>
```

```
	<xs:complexType name="fixedRecordEncodingType">
		<xs:simpleContent>
			<xs:extension base="fixedRecordEncodingUnion">
				<xs:attributeGroup ref="modelID:commonAttributes"/>
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>
	<xs:simpleType name="variantRecordEnumeratorPattern">
		<xs:restriction base="xs:NCName">
			<xs:pattern value="HLAother|(\S)+|([(\S)+..(\S)+])"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="variantRecordEncodingEnumerator">
		<xs:restriction base="xs:string">
			<xs:enumeration value="HLAvariantRecord"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="variantRecordEncodingUnion">
		<xs:union memberTypes="variantRecordEncodingEnumerator modelID:nonEmptyString"/>
	</xs:simpleType>
	<xs:complexType name="variantRecordEncodingType">
		<xs:simpleContent>
			<xs:extension base="variantRecordEncodingUnion">
				<xs:attributeGroup ref="modelID:commonAttributes"/>
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>
</xs:schema>
```

This page left intentionally blank

Annex B – BOM Example

The XML listing of the BOM Restaurant Payment example highlighted previously in this document, which conforms to the schemas identified in Annex A, is provided below.

BOM Restaurant Payment

```
<?xml version="1.0"?>
<!-- edited with XMLSpy v2005 U (http://www.xmlspy.com) by Tram Chase (SimVentions) -->
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com) by Bjorn Lofstrand (Pitch Technologies) -->
<!-- Example BOM. The only namespaces needed apart from the standards-->
<BOM xmlns="http://www.sisostds.org/schemas/bom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:omt="http://www.sisostds.org/schemas/IEEE1516.2-2006" xmlns:modelID="http://www.sisostds.org/schemas/modelID"
xsi:schemaLocation="http://www.sisostds.org/schemas/bom BOM_v2006.xsd">
	<modelIdentification>
		<modelID:name>RestaurantPayment</modelID:name>
		<modelID:type>BOM</modelID:type>
		<modelID:version>1.0 Beta</modelID:version>
		<modelID:modificationDate>2006-03-21</modelID:modificationDate>
		<modelID:securityClassification>Unclassified</modelID:securityClassification>
		<modelID:releaseRestriction>Not for release outside the XYZ Restaurant Corporation.</modelID:releaseRestriction>
		<modelID:releaseRestriction>Release only to BOM PDG members.</modelID:releaseRestriction>
		<modelID:purpose> Standardize the pattern of interaction between the customer and waiter for payment of meal at
restaurant.</modelID:purpose>
		<modelID:applicationDomain>Restaurant operations</modelID:applicationDomain>
		<modelID:description>This is an example to illustrate the key concepts of an interface BOM.</modelID:description>
		<modelID:useLimitation>Not applicable to drive-through operations</modelID:useLimitation>
		<modelID:useHistory>Utilized in the Restaurant FOM</modelID:useHistory>
		<modelID:keyword>
			<modelID:taxonomy>Restaurant taxonomy</modelID:taxonomy>
			<modelID:keywordValue>Restaurant</modelID:keywordValue>
		</modelID:keyword>
		<modelID:keyword>
			<modelID:taxonomy>Commerce taxonomy</modelID:taxonomy>
			<modelID:keywordValue>Payment</modelID:keywordValue>
		</modelID:keyword>
		<modelID:poc>
			<modelID:pocType>Primary author</modelID:pocType>
			<modelID:pocName>Mr. Snuffy Smith</modelID:pocName>
			<modelID:pocOrg>XYZ Restaurant Corporation</modelID:pocOrg>
			<modelID:pocTelephone>+1 44 123-456-7890</modelID:pocTelephone>
			<modelID:pocEmail>snuffy.smith@goodeatscafe.com</modelID:pocEmail>
		</modelID:poc>
		<modelID:poc>
			<modelID:pocType>Release authority</modelID:pocType>
			<modelID:pocOrg>XYZ Restaurant Corporation</modelID:pocOrg>
			<modelID:pocTelephone>+1 44 123-456-1000</modelID:pocTelephone>
			<modelID:pocEmail>the.lawyer@xyz-foods.com</modelID:pocEmail>
		</modelID:poc>
		<modelID:reference>
			<modelID:type>Glossary</modelID:type>
			<modelID:identification>ISBN 12345678901</modelID:identification>
		</modelID:reference>
		<modelID:reference>
			<modelID:type>Conceptual Model</modelID:type>
			<modelID:identification>http://boms.info/resaurantconceptualmodel.doc</modelID:identification>
		</modelID:reference>
		<modelID:other>This BOM pattern of interplay was featured on the Food Network special “HLA and the Future of Restaurant
Operations”</modelID:other>
	</modelIdentification>
	<conceptualModelDefinition>
		<patternInterplay>
			<name>RestaurantPayment</name>
			<action>
				<sequence>1</sequence>
				<name>CustomerDone</name>
				<sender>CustomerEntity</sender>
				<receiver>WaiterEntity</receiver>
				<semantics>Waiter is made aware that the customer will not order anything else</semantics>
				<variation>
					<name>CustomerIdle</name>
					<event>DirtyDishesOnTable</event>
					<sender>TableEntity</sender>
					<receiver>WaiteEntityr</receiver>
					<semantics>Waiter observes that the table is full of dirty dishes</semantics>
```

```
                    </variation>
                    <variation>
                        <name>CustomerRequest</name>
                        <event>CheckRequest</event>
                        <sender>CustomerEntity</sender>
                        <receiver>WaiterEntity</receiver>
                        <semantics>Waiter is called by customer</semantics>
                    </variation>
                </action>
                <action>
                    <sequence>2</sequence>
                    <name>CheckBroughtToTable</name>
                    <event>PaymentDueNotification</event>
                    <sender>WaiterEntity</sender>
                    <receiver>CustomerEntity</receiver>
                    <semantics>Waiter notifies customer of payment due</semantics>
                </action>
                <action>
                    <sequence>3</sequence>
                    <name>CustomerPays</name>
                    <sender>CustomerEntity</sender>
                    <receiver>WaiterEntity</receiver>
                    <variation>
                        <name>BillPayed_Cash</name>
                        <bom>CashPaymentBOM</bom>
                        <sender>CustomerEntity</sender>
                        <receiver>WaiterEntity</receiver>
                    </variation>
                    <variation>
                        <name>BillPayed_Credit</name>
                        <bom>CreditCardPaymentBOM</bom>
                        <sender>CustomerEntity</sender>
                        <receiver>WaiterEntity</receiver>
                    </variation>
                    <exception>
                        <name>BillNotPayed</name>
                        <event>NonPayingCustomerTrigger</event>
                        <sender>CustomerEntity</sender>
                        <condition>Customer leaves without paying bill</condition>
                    </exception>
                </action>
                <action>
                    <sequence>4</sequence>
                    <name>CustomerLeavesTip</name>
                    <event>Tip</event>
                    <sender>CustomerEntity</sender>
                    <receiver>WaiterEntity</receiver>
                    <semantics>Waiter notifies customer of payment due</semantics>
                </action>
            </patternInterplay>
            <stateMachine>
                <name>EmployeeSM</name>
                <conceptualEntity>Greeter</conceptualEntity>
                <conceptualEntity>Waiter</conceptualEntity>
                <state>
                    <name>Ready</name>
                    <exitCondition>
                        <exitAction>RestaurantVisitationBOM.CustomerArrives</exitAction>
                        <nextState>Greet</nextState>
                    </exitCondition>
                </state>
                <state>
                    <name>Greet</name>
                    <exitCondition>
                        <exitAction>RestaurantVisitationBOM.GreetCustomer</exitAction>
                        <nextState>Seat</nextState>
                    </exitCondition>
                </state>
                <state>
                    <name>Seat</name>
                    <exitCondition>
                        <exitAction>RestaurantVisitationBOM.SeatCustomer</exitAction>
                        <nextState>TakeOrder</nextState>
                    </exitCondition>
                </state>
                <state>
                    <name>MonitorCustomer</name>
                    <exitCondition>
                        <exitAction>RestaurantFoodPreparationBOM.CustomerOrders</exitAction>
```

```
                        <nextState>ProcessOrder</nextState>
                    </exitCondition>
                    <exitCondition>
                        <exitAction>RestaurantPaymentBOM.CustomerRequestsBill</exitAction>
                        <nextState>PrepareBill</nextState>
                    </exitCondition>
                    <exitCondition>
                        <exitAction>RestaurantPaymentBOM.CustomerPays</exitAction>
                        <nextState>ProcessBill</nextState>
                    </exitCondition>
                    <exitCondition>
                        <exitAction>RestaurantVisitationBOM.CustomerLeft</exitAction>
                        <nextState>ClearingTable</nextState>
                    </exitCondition>
                </state>
                <state>
                    <name>ProcessOrder</name>
                    <exitCondition>
                        <exitAction>RestaurantFoodPreparationBOM.SubmitOrderToKitchen</exitAction>
                        <nextState>MonitorKitchenOrder</nextState>
                    </exitCondition>
                    <exitCondition>
                        <exitAction>RestaurantFoodPreparationBOM.SubmitOrderToKitchen</exitAction>
                        <nextState>MonitorCustomer</nextState>
                    </exitCondition>
                </state>
                <state>
                    <name>MonitorKitchenOrder</name>
                    <exitCondition>
                        <exitAction>RestaurantFoodPreparationBOM.FoodReadyToServe</exitAction>
                        <nextState>Serve</nextState>
                    </exitCondition>
                </state>
                <state>
                    <name>Serve</name>
                    <exitCondition>
                        <exitAction>RestaurantFoodPreparationBOM.FoodBroughtToTable</exitAction>
                        <nextState>MonitorCustomer</nextState>
                    </exitCondition>
                </state>
                <state>
                    <name>PrepareBill</name>
                    <exitCondition>
                        <exitAction>RestaurantPaymentBOM.CheckBroughtToTable</exitAction>
                        <nextState>MonitorCustomer</nextState>
                    </exitCondition>
                </state>
                <state>
                    <name>ProcessBill</name>
                    <exitCondition>
                        <exitAction>RestaurantPaymentBOM.ReceiptChangeReturned</exitAction>
                        <nextState>MonitorCustomer</nextState>
                    </exitCondition>
                </state>
                <state>
                    <name>ClearingTable</name>
                    <exitCondition>
                        <exitAction>RestaurantPaymentBOM.ReceiptChangeReturned</exitAction>
                        <nextState>Ready</nextState>
                    </exitCondition>
                </state>
            </stateMachine>
            <stateMachine>
                <name>CustomerSM</name>
                <conceptualEntity>Customer</conceptualEntity>
                <state>
                    <name>Eating</name>
                    <exitCondition>
                        <exitAction>RestaurantPaymentBOM.CustomerDone</exitAction>
                        <nextState>Ready</nextState>
                    </exitCondition>
                </state>
                <state>
                    <name>Ready</name>
                    <exitCondition>
                        <exitAction>RestaurantPaymentBOM.CheckBroughtToTable</exitAction>
                        <nextState>Pay</nextState>
                    </exitCondition>
                </state>
```

```
		<state>
			<name>Pay</name>
			<exitCondition>
				<exitAction>RestaurantPaymentBOM.CustomerPays</exitAction>
				<nextState>Ready</nextState>
			</exitCondition>
		</state>
	</stateMachine>
	<entityTypes>
		<entityType>
			<name>TableEntity</name>
			<characteristic>
				<name>ID</name>
			</characteristic>
			<characteristic>
				<name>Occupied</name>
			</characteristic>
			<characteristic>
				<name>Dishes_Dirty</name>
			</characteristic>
		</entityType>
		<entityType>
			<name>CustomerEntity</name>
			<characteristic>
				<name>ID</name>
			</characteristic>
			<characteristic>
				<name>Table_ID</name>
			</characteristic>
			<characteristic>
				<name>FoodOrder</name>
			</characteristic>
			<characteristic>
				<name>PaidBill</name>
			</characteristic>
		</entityType>
		<entityType>
			<name>WaiterEntity</name>
			<characteristic>
				<name>ID</name>
			</characteristic>
			<characteristic>
				<name>Assigned_Tables</name>
			</characteristic>
		</entityType>
		<entityType>
			<name>BillEntity</name>
			<characteristic>
				<name>Amount</name>
			</characteristic>
			<characteristic>
				<name>Tip</name>
			</characteristic>
			<characteristic>
				<name>Waiter_ID</name>
			</characteristic>
			<characteristic>
				<name>Table_ID</name>
			</characteristic>
		</entityType>
	</entityTypes>
	<eventTypes>
		<eventType>
			<name>DirtyDishesOnTable</name>
			<sourceCharacteristic>
				<name>TableEntity.ID</name>
			</sourceCharacteristic>
			<triggerCondition>TableEntity.Dishes_Dirty = TRUE</triggerCondition>
			<semantics>Trigger when there is dirty dishes on the table</semantics>
		</eventType>
		<eventType>
			<name>NonPayingCustomer</name>
			<sourceCharacteristic>
				<name>CustomerEntity.ID</name>
			</sourceCharacteristic>
			<triggerCondition>CustomerEntity.PaidBill = FALSE</triggerCondition>
			<semantics>Trigger when customer leaves without paying</semantics>
		</eventType>
		<eventType>
```

```
                        <name>CheckRequest</name>
                        <sourceCharacteristic>
                                <name>CustomerEntity.ID</name>
                        </sourceCharacteristic>
                        <targetCharacteristic>
                                <name>WaiterEntity.ID</name>
                        </targetCharacteristic>
                        <semantics>Request check from waiter</semantics>
                </eventType>
                <eventType>
                        <name>PaymentDueNotification</name>
                        <sourceCharacteristic>
                                <name>WaiterEntity.ID</name>
                        </sourceCharacteristic>
                        <targetCharacteristic>
                                <name>CustomerEntity.ID</name>
                        </targetCharacteristic>
                        <contentCharacteristic>
                                <name>BillEntity.Amount</name>
                        </contentCharacteristic>
                        <semantics>Notificatition of Payment Due</semantics>
                </eventType>
                <eventType>
                        <name>Tip</name>
                        <sourceCharacteristic>
                                <name>CustomerEntity.ID</name>
                        </sourceCharacteristic>
                        <targetCharacteristic>
                                <name>WaiterEntity.ID</name>
                        </targetCharacteristic>
                        <contentCharacteristic>
                                <name>BillEntity.Tip</name>
                        </contentCharacteristic>
                        <semantics>Tip left for waiter by customer</semantics>
                </eventType>
        </eventTypes>
</conceptualModelDefinition>
<modelMapping>
        <entityTypeMappings>
                <entityTypeMapping>
                        <name>TableEntity</name>
                        <hlaClass>HLAobjectRoot.PhysicalEntity.Table</hlaClass>
                        <characteristic>
                                <name>ID</name>
                                <hlaProperty>Table.ID</hlaProperty>
                        </characteristic>
                        <characteristic>
                                <name>Occupied</name>
                                <hlaProperty>Table.isOccupied</hlaProperty>
                        </characteristic>
                        <characteristic>
                                <name>Dishes_Dirty</name>
                                <hlaProperty>Table.hasDirtyDishes</hlaProperty>
                        </characteristic>
                </entityTypeMapping>
                <entityTypeMapping>
                        <name>CustomerEntity</name>
                        <hlaClass>HLAobjectRoot.PhysicalEntity.Human.Customer</hlaClass>
                        <characteristic>
                                <name>ID</name>
                                <hlaProperty>Customer.ID</hlaProperty>
                        </characteristic>
                        <characteristic>
                                <name>Table_ID</name>
                                <hlaProperty>Customer.AssignedTableID</hlaProperty>
                        </characteristic>
                        <characteristic>
                                <name>FoodOrder</name>
                                <hlaProperty>Customer.Order</hlaProperty>
                        </characteristic>
                        <characteristic>
                                <name>PaidBill</name>
                                <hlaProperty>Customer.hasPaid</hlaProperty>
                        </characteristic>
                </entityTypeMapping>
                <entityTypeMapping>
                        <name>WaiterEntity</name>
                        <hlaClass>HLAobjectRoot.PhysicalEntity.Human.Waiter</hlaClass>
                        <characteristic>
```

```
                        <name>Name</name>
                        <hlaProperty>Waiter.ID</hlaProperty>
                    </characteristic>
                    <characteristic>
                        <name>Assigned_Tables</name>
                        <hlaProperty>Waiter.AssignedTables[]</hlaProperty>
                    </characteristic>
                </entityTypeMapping>
                <entityTypeMapping>
                    <name>BillEntity</name>
                    <hlaClass>HLAobjectRoot.PhysicalEntity.Payment</hlaClass>
                    <hlaClass>HLAobjectRoot.PhysicalEntity.Human.Waiter</hlaClass>
                    <hlaClass>HLAobjectRoot.PhysicalEntity.Human.Customer</hlaClass>
                    <characteristic>
                        <name>Amount</name>
                        <hlaProperty>Payment.Amount</hlaProperty>
                    </characteristic>
                    <characteristic>
                        <name>Tip</name>
                        <hlaProperty>Payment.Tip</hlaProperty>
                    </characteristic>
                    <characteristic>
                        <name>Waiter_ID</name>
                        <hlaProperty>Waiter.ID</hlaProperty>
                    </characteristic>
                    <characteristic>
                        <name>Table_ID</name>
                        <hlaProperty>Customer.AssignedTableID</hlaProperty>
                    </characteristic>
                </entityTypeMapping>
            </entityTypeMappings>
            <eventTypeMappings>
                <eventTypeMapping>
                    <name>DirtyDishesOnTable</name>
                    <hlaClass>HLAobjectRoot.PhysicalEntity.Table</hlaClass>
                    <sourceCharacteristic>
                        <name>TableEntity.ID</name>
                        <hlaProperty>Table.ID</hlaProperty>
                    </sourceCharacteristic>
                    <triggerCondition>
                        <name>TableEntity.Dishes_Dirty_equals_TRUE</name>
                        <hlaProperty>Table.hasDirtyDishes==TRUE</hlaProperty>
                    </triggerCondition>
                </eventTypeMapping>
                <eventTypeMapping>
                    <name>NonPayingCustomer</name>
                    <hlaClass>HLAinteractionRoot.PhysicalEvent.CustomerLeaves</hlaClass>
                    <hlaClass>HLAobjectRoot.PhysicalEntity.Customer</hlaClass>
                    <sourceCharacteristic>
                        <name>TableEntity.ID</name>
                        <hlaProperty>HLAobjectRoot.PhysicalEntity.Table.ID</hlaProperty>
                    </sourceCharacteristic>
                    <triggerCondition>
                        <name>TableEntity.Dishes_Dirty_equals_TRUE</name>
                        <hlaProperty>Table.hasDirtyDishes == TRUE</hlaProperty>
                    </triggerCondition>
                </eventTypeMapping>
                <eventTypeMapping>
                    <name>CheckRequest</name>
                    <hlaClass>HLAinteractionRoot.RealWorldMessage.RequestBill</hlaClass>
                    <sourceCharacteristic>
                        <name>CustomerEntity.ID</name>
                        <hlaProperty>RequestBill.CustomerID</hlaProperty>
                    </sourceCharacteristic>
                    <targetCharacteristic>
                        <name>WaiterEntity.ID</name>
                        <hlaProperty>RequestBill.WaiterID</hlaProperty>
                    </targetCharacteristic>
                </eventTypeMapping>
                <eventTypeMapping>
                    <name>PaymentDueNotification</name>
                    <hlaClass>HLAinteractionRoot.RealWorldMessage.PaymentDue</hlaClass>
                    <sourceCharacteristic>
                        <name>WaiterEntity.ID</name>
                        <hlaProperty>PaymentDue.WaiterID</hlaProperty>
                    </sourceCharacteristic>
                    <targetCharacteristic>
                        <name>CustomerEntity.ID</name>
                        <hlaProperty>PaymentDue.CustomerID</hlaProperty>
```

```
                        </targetCharacteristic>
                        <contentCharacteristic>
                                <name>BillEntity.Amount</name>
                                <hlaProperty>HLAinteractionRoot.RealWorldMessage.PaymentDue.Amount</hlaProperty>
                        </contentCharacteristic>
                </eventTypeMapping>
                <eventTypeMapping>
                        <name>Tip</name>
                        <hlaClass>HLAobjectRoot.PhysicalEntity.Payment</hlaClass>
                        <sourceCharacteristic>
                                <name>CustomerEntity.ID</name>
                                <hlaProperty>Payment.Source</hlaProperty>
                        </sourceCharacteristic>
                        <targetCharacteristic>
                                <name>WaiterEntity.ID</name>
                                <hlaProperty>Payment.Recipient</hlaProperty>
                        </targetCharacteristic>
                        <contentCharacteristic>
                                <name>BillEntity.Tip</name>
                                <hlaProperty>Payment.Tip</hlaProperty>
                        </contentCharacteristic>
                </eventTypeMapping>
        </eventTypeMappings>
</modelMapping>
<objectModelDefinition>
        <objects>
                <omt:objectClass>
                        <omt:name>HLAobjectRoot</omt:name>
                        <omt:sharing>Neither</omt:sharing>
                        <omt:attribute>
                                <omt:name>HLApriviledgeToDeleteObject</omt:name>
                                <omt:dataType>HLAboolean</omt:dataType>
                                <omt:updateType>Static</omt:updateType>
                                <omt:ownership>NoTransfer</omt:ownership>
                                <omt:sharing>Subscribe</omt:sharing>
                                <omt:transportation>HLAreliable</omt:transportation>
                                <omt:order>TimeStamp</omt:order>
                        </omt:attribute>
                        <omt:objectClass>
                                <omt:name>PhysicalEntity</omt:name>
                                <omt:sharing>PublishSubscribe</omt:sharing>
                                <omt:attribute>
                                        <omt:name>ID</omt:name>
                                        <omt:dataType>ID</omt:dataType>
                                        <omt:updateType>Static</omt:updateType>
                                        <omt:ownership>DivestAcquire</omt:ownership>
                                        <omt:sharing>PublishSubscribe</omt:sharing>
                                        <omt:transportation>HLAreliable</omt:transportation>
                                        <omt:order>Receive</omt:order>
                                </omt:attribute>
                                <omt:objectClass>
                                        <omt:name>Table</omt:name>
                                        <omt:sharing>PublishSubscribe</omt:sharing>
                                        <omt:attribute>
                                                <omt:name>hasDirtyDishes</omt:name>
                                                <omt:dataType>Boolean</omt:dataType>
                                                <omt:updateType>Conditional</omt:updateType>
                                                <omt:updateCondition>onChange or request</omt:updateCondition>
                                                <omt:ownership>DivestAcquire</omt:ownership>
                                                <omt:sharing>PublishSubscribe</omt:sharing>
                                                <omt:transportation>HLAreliable</omt:transportation>
                                                <omt:order>Receive</omt:order>
                                        </omt:attribute>
                                        <omt:attribute>
                                                <omt:name>isOccupied</omt:name>
                                                <omt:dataType>Boolean</omt:dataType>
                                                <omt:updateType>Conditional</omt:updateType>
                                                <omt:updateCondition>onChange or request</omt:updateCondition>
                                                <omt:ownership>DivestAcquire</omt:ownership>
                                                <omt:sharing>PublishSubscribe</omt:sharing>
                                                <omt:transportation>HLAreliable</omt:transportation>
                                                <omt:order>Receive</omt:order>
                                        </omt:attribute>
                                </omt:objectClass>
                                <omt:objectClass>
                                        <omt:name>Customer</omt:name>
                                        <omt:sharing>PublishSubscribe</omt:sharing>
                                        <omt:attribute>
                                                <omt:name>TableID</omt:name>
```

```
                                        <omt:dataType>ID</omt:dataType>
                                        <omt:updateType>Static</omt:updateType>
                                        <omt:ownership>DivestAcquire</omt:ownership>
                                        <omt:sharing>PublishSubscribe</omt:sharing>
                                        <omt:transportation>HLAreliable</omt:transportation>
                                        <omt:order>Receive</omt:order>
                                </omt:attribute>
                                <omt:attribute>
                                        <omt:name>hasPaid</omt:name>
                                        <omt:dataType>Boolean</omt:dataType>
                                        <omt:updateType>Conditional</omt:updateType>
                                        <omt:updateCondition>onChange or request</omt:updateCondition>
                                        <omt:ownership>DivestAcquire</omt:ownership>
                                        <omt:sharing>PublishSubscribe</omt:sharing>
                                        <omt:transportation>HLAreliable</omt:transportation>
                                        <omt:order>Receive</omt:order>
                                </omt:attribute>
                                <omt:attribute>
                                        <omt:name>Order</omt:name>
                                        <omt:dataType>OrderType</omt:dataType>
                                        <omt:updateType>Conditional</omt:updateType>
                                        <omt:updateCondition>onChange or request</omt:updateCondition>
                                        <omt:ownership>DivestAcquire</omt:ownership>
                                        <omt:sharing>PublishSubscribe</omt:sharing>
                                        <omt:transportation>HLAreliable</omt:transportation>
                                        <omt:order>Receive</omt:order>
                                </omt:attribute>
                        </omt:objectClass>
                        <omt:objectClass>
                                <omt:name>Waiter</omt:name>
                                <omt:sharing>PublishSubscribe</omt:sharing>
                                <omt:attribute>
                                        <omt:name>AssignedTables</omt:name>
                                        <omt:dataType>AssignedTablesType</omt:dataType>
                                        <omt:updateType>Conditional</omt:updateType>
                                        <omt:updateCondition>onChange or request</omt:updateCondition>
                                        <omt:ownership>DivestAcquire</omt:ownership>
                                        <omt:sharing>PublishSubscribe</omt:sharing>
                                        <omt:transportation>HLAreliable</omt:transportation>
                                        <omt:order>Receive</omt:order>
                                </omt:attribute>
                        </omt:objectClass>
                </omt:objectClass>
                <omt:objectClass>
                        <omt:name>AbstractEntity</omt:name>
                        <omt:sharing>PublishSubscribe</omt:sharing>
                        <omt:objectClass>
                                <omt:name>Payment</omt:name>
                                <omt:sharing>PublishSubscribe</omt:sharing>
                                <omt:semantics>Represents a payment using cash.</omt:semantics>
                                <omt:attribute>
                                        <omt:name>Amount</omt:name>
                                        <omt:dataType>Float32BE</omt:dataType>
                                        <omt:updateType>Conditional</omt:updateType>
                                        <omt:updateCondition>on change or request</omt:updateCondition>
                                        <omt:ownership>DivestAcquire</omt:ownership>
                                        <omt:sharing>PublishSubscribe</omt:sharing>
                                        <omt:transportation>HLAreliable</omt:transportation>
                                        <omt:order>Receive</omt:order>
                                        <omt:semantics>Represents a payment using cash."</omt:semantics>
                                </omt:attribute>
                                <omt:attribute>
                                        <omt:name>Tip</omt:name>
                                        <omt:dataType>Float32BE</omt:dataType>
                                        <omt:updateType>Conditional</omt:updateType>
                                        <omt:updateCondition>on change or request</omt:updateCondition>
                                        <omt:ownership>DivestAcquire</omt:ownership>
                                        <omt:sharing>PublishSubscribe</omt:sharing>
                                        <omt:transportation>HLAreliable</omt:transportation>
                                        <omt:order>Receive</omt:order>
                                        <omt:semantics>Represents a payment using cash."</omt:semantics>
                                </omt:attribute>
                                <omt:attribute>
                                        <omt:name>Source</omt:name>
                                        <omt:dataType>ID</omt:dataType>
                                        <omt:updateType>Static</omt:updateType>
                                        <omt:ownership>DivestAcquire</omt:ownership>
                                        <omt:sharing>PublishSubscribe</omt:sharing>
                                        <omt:transportation>HLAreliable</omt:transportation>
```

```
                                        <omt:order>Receive</omt:order>
                                        <omt:semantics>Represents a payment using cash.</omt:semantics>
                                </omt:attribute>
                                <omt:attribute>
                                        <omt:name>Recipient</omt:name>
                                        <omt:dataType>ID</omt:dataType>
                                        <omt:updateType>Static</omt:updateType>
                                        <omt:ownership>DivestAcquire</omt:ownership>
                                        <omt:sharing>PublishSubscribe</omt:sharing>
                                        <omt:transportation>HLAreliable</omt:transportation>
                                        <omt:order>Receive</omt:order>
                                        <omt:semantics>Represents a payment using cash.</omt:semantics>
                                </omt:attribute>
                        </omt:objectClass>
                </omt:objectClass>
        </omt:objectClass>
</objects>
<interactions>
        <omt:interactionClass>
                <omt:name>HLAinteractionRoot</omt:name>
                <omt:sharing>Subscribe</omt:sharing>
                <omt:transportation>HLAreliable</omt:transportation>
                <omt:order>Receive</omt:order>
                <omt:interactionClass>
                        <omt:name>PhysicalEvent</omt:name>
                        <omt:sharing>PublishSubscribe</omt:sharing>
                        <omt:transportation>HLAreliable</omt:transportation>
                        <omt:order>Receive</omt:order>
                        <omt:parameter>
                                <omt:name>CustomerID</omt:name>
                                <omt:dataType>ID</omt:dataType>
                        </omt:parameter>
                        <omt:interactionClass>
                                <omt:name>CustomerLeaves</omt:name>
                                <omt:sharing>PublishSubscribe</omt:sharing>
                                <omt:transportation>HLAreliable</omt:transportation>
                                <omt:order>Receive</omt:order>
                        </omt:interactionClass>
                </omt:interactionClass>
                <omt:interactionClass>
                        <omt:name>RealWorldMessage</omt:name>
                        <omt:sharing>PublishSubscribe</omt:sharing>
                        <omt:transportation>HLAreliable</omt:transportation>
                        <omt:order>Receive</omt:order>
                        <omt:parameter>
                                <omt:name>CustomerID</omt:name>
                                <omt:dataType>ID</omt:dataType>
                        </omt:parameter>
                        <omt:interactionClass>
                                <omt:name>RequestBill</omt:name>
                                <omt:sharing>PublishSubscribe</omt:sharing>
                                <omt:transportation>HLAreliable</omt:transportation>
                                <omt:order>Receive</omt:order>
                                <omt:semantics>Customer request check from waiter</omt:semantics>
                                <omt:parameter>
                                        <omt:name>WaiterID</omt:name>
                                        <omt:dataType>ID</omt:dataType>
                                </omt:parameter>
                        </omt:interactionClass>
                        <omt:interactionClass>
                                <omt:name>PaymentDue</omt:name>
                                <omt:sharing>PublishSubscribe</omt:sharing>
                                <omt:transportation>HLAreliable</omt:transportation>
                                <omt:order>Receive</omt:order>
                                <omt:semantics>Customer request check from waiter</omt:semantics>
                                <omt:parameter>
                                        <omt:name>Amount</omt:name>
                                        <omt:dataType>Float32BE</omt:dataType>
                                </omt:parameter>
                                <omt:parameter>
                                        <omt:name>WaiterID</omt:name>
                                        <omt:dataType>ID</omt:dataType>
                                </omt:parameter>
                        </omt:interactionClass>
                </omt:interactionClass>
        </omt:interactionClass>
</interactions>
<dataTypes>
        <omt:basicDataRepresentations>
```

```
                                    <omt:basicData>
                                        <omt:name>HLAoctet</omt:name>
                                        <omt:size>8</omt:size>
                                        <omt:interpretation>8-bit value</omt:interpretation>
                                        <omt:endian>Big</omt:endian>
                                        <omt:encoding>Assumed to be portable among hardware devices</omt:encoding>
                                    </omt:basicData>
                                    <omt:basicData>
                                        <omt:name>HLAinteger16BE</omt:name>
                                        <omt:size>16</omt:size>
                                        <omt:interpretation>Integer in the range [?2^15, 2^15 ? 1].</omt:interpretation>
                                        <omt:endian>Big</omt:endian>
                                        <omt:encoding>16-bit two?s complement signed integer. The most significant bit contains the
sign.</omt:encoding>
                                    </omt:basicData>
                                    <omt:basicData>
                                        <omt:name>HLAfloat32BE</omt:name>
                                        <omt:size>32</omt:size>
                                        <omt:interpretation>Single-precision floating-point number.</omt:interpretation>
                                        <omt:endian>Big</omt:endian>
                                        <omt:encoding>32-bit IEEE normalized single-precision format (see IEEE Std. 754-
1985).</omt:encoding>
                                    </omt:basicData>
                                </omt:basicDataRepresentations>
                                <omt:simpleDataTypes>
                                    <omt:simpleData>
                                        <omt:name>ID</omt:name>
                                        <omt:representation>HLAinteger16BE</omt:representation>
                                    </omt:simpleData>
                                </omt:simpleDataTypes>
                                <omt:enumeratedDataTypes></omt:enumeratedDataTypes>
                                <omt:arrayDataTypes>
                                    <omt:arrayData>
                                        <omt:name>AssignedTablesType</omt:name>
                                        <omt:dataType>ID</omt:dataType>
                                        <omt:cardinality>99</omt:cardinality>
                                        <omt:encoding>BigEndian</omt:encoding>
                                    </omt:arrayData>
                                </omt:arrayDataTypes>
                                <omt:fixedRecordDataTypes>
                                    <omt:fixedRecordData>
                                        <omt:name>OrderType</omt:name>
                                        <omt:encoding>HLAfixedRecord</omt:encoding>
                                        <omt:field>
                                            <omt:name>Item</omt:name>
                                            <omt:dataType>HLAinteger16BE</omt:dataType>
                                        </omt:field>
                                        <omt:field>
                                            <omt:name>Quantity</omt:name>
                                            <omt:dataType>HLAinteger16BE</omt:dataType>
                                        </omt:field>
                                    </omt:fixedRecordData>
                                </omt:fixedRecordDataTypes>
                                <omt:variantRecordDataTypes></omt:variantRecordDataTypes>
                            </dataTypes>
                        </objectModelDefinition>
                        <notes>
                            <omt:note>
                                <omt:label>ID1</omt:label>
                                <omt:semantics>This pattern of interplay might be able to be used as a template to support payment patterns for other
service oriented businesses</omt:semantics>
                            </omt:note>
                            <omt:note>
                                <omt:label>ID2</omt:label>
                                <omt:semantics>This pattern action may involve other employees including the host (i.e. Greeter) </omt:semantics>
                            </omt:note>
                            <omt:note>
                                <omt:label>ID3</omt:label>
                                <omt:semantics>The support for this trigger may involve a completely other BOM  (pattern) to support non-paying
customer</omt:semantics>
                            </omt:note>
                            <omt:note>
                                <omt:label>ID4</omt:label>
                                <omt:semantics>Customer may also request check from other employee conceptual entity types</omt:semantics>
                            </omt:note>
                            <omt:note>
                                <omt:label>ID5</omt:label>
                                <omt:semantics>This is the idle state for the Greeter or Waiter</omt:semantics>
                            </omt:note>
```

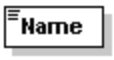
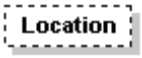
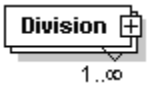
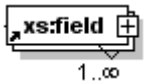
```
        </notes>
</BOM>
```

This page left intentionally blank

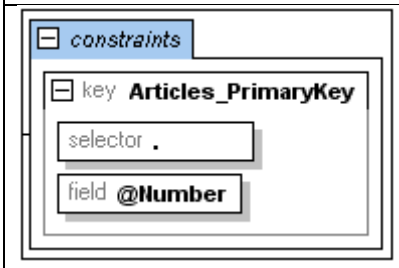
Annex C – XMLSpy® Schema Design Content Model

Many of the XML schema related diagrams provided within this specification (see Section 7) were generated using the Altova® XMLSpy® tool. This section describes the symbols and nomenclature found within these diagrams beginning with the Element Symbols identified in Table C-1.[6]

Table C-1 Element Symbols

Symbol	Name	Description
	Mandatory single element	A rectangle with a solid border identfies a required XML schema element. In this example, a mandatory element of `Country` is required to be identified within the XML instance data.
	Mandatory single element containing parsed character data (i.e., a child text node)	A rectangle with a solid border and line markings in the upper left corner identifies a mandatory XML schema element used to provide simple content (text node only) or mixed content (text and child elements).I n this example, simple content is denoted because there is no plus sign; a mandatory element of `Name` containing simple content is defined. This `Name` element with simple content would be required to be identified within the XML instance data.
	Single optional element	A rectangle with a dashed border identifies an optional XML schema element. In this example, an optional element of `Location` may be identified within the XML instance data.
	Mandatory multiple element	An overlapping set of rectangles identifies a mandatory XML schema element represented by one or more instances. In this example, up to five `Alias` element values may be identified within the XML instance data.
	Mandatory multiple element containing child elements	A mandatory multiple element with a plus sign identifies an element value containing child elements. In this example, an unlimited number of `Division` element values may be defined with child elements within the XML instance data.
	Element referencing global element	The arrow in the bottom left indicates an element referencing a global element, which is defined elsewhere. In this example, an unlimited number of `xs:field` element values may be defined with child elements within the XML instance data.
	Complex Type	An irregular hexagon with a plus sign indicates a complex data type. Complex types can be used either as (i) the datatype of an element, or (ii) the base type of another complex type. In this example, `keybase` is a global complex type used to define other aspects of the XML schema.
	Model Group	An irregular octagon with a plus sign indicates a model group, which can be used to define and reuse element declarations. In this example, `Subsidaries` is a global model group that can be used to define elements within the XML schema. reuse element declarations.
	Wildcards	An irregular octagon with `any` at left indicates a wildcard, which can be used as placeholders to allow elements not specified in the schema or from other namespaces. The common wildcards used within an XML schema are identified below:

[6] "Introduction to XMLSPY 2005 Enterprise Edition," http://www.altova.com/manual/

		`##other` = elements that can belong to any namespace other than the target namespace defined in the schema; `##any` = elements that can belong to any namespace; `##targetNamespace` = elements that must belong to the target namespace defined in the schema; `##local` = elements that cannot belong to any namespace; *`anyURI`* = elements that belong to the namespace you specify.
	Attributes	A double lined rectangle with the word *'attributes'* in italics, which can be expanded, is used to define XML attributes. Within the interior of the double line rectangle, each attribute is shown in a rectangle with a dashed border. Attributes are of the possession of an element. In this example, a `href` is the only attribute defined and its owning element is not shown.
	Identity constraints	A rectangle with the word '*constraints*' in italics is used to define constraints of a content model. The identity constraints listed in the content model of a component show constraints as defined with the `key` and `keyref` elements, and with the `unique` element. In this example, Articles_PrimaryKey is defined as a key constraint where the field to be constrained is `Number` with the selector of ".", which is a decimal point and used to verify the `Number` is valid.

Simple types

A "simple type" element is defined as a data type that only contains values and no element or attributes. For instance, the element type may be prefixed by the namespace prefix **xsd: string**, indicating that it is a predefined XML Schema data type. An example is illustrated below:

In this example, the name for the "simple type" is **Name**, and the type used to define **Name** is **xsd:string**.

Complex types

"Complex type" is a data type that may contain attributes, elements, and text. Adding sub-elements to an element, automatically defines the element with the content model as complex. An example is illustrated in Figure C-1.

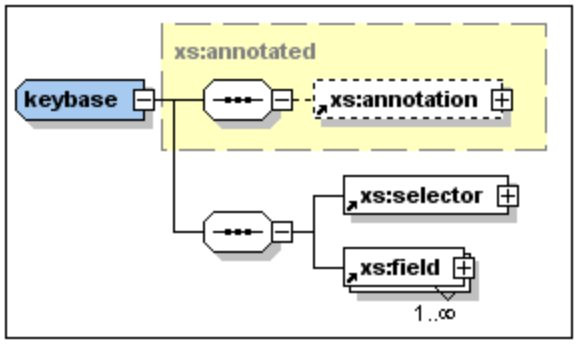


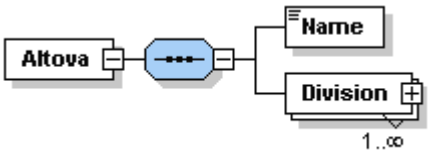
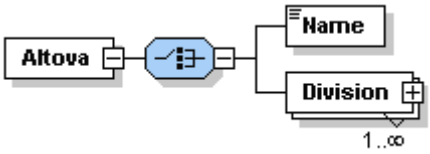
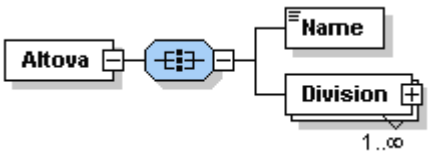
Figure C-1 Complex Types

The **keybase** complex type shown in Figure C-1 was declared with a base type of **xs:annotated**. The base type is displayed as a rectangle with a dashed gray border and a yellow background color. The child elements **xs:selector** and **xs:field** extend upon the base type. (Note the tiny arrows in the bottom left corner of the **xs:selector** and **xs:field** rectangles. These indicate that both elements reference global elements of those names.)

Compositors

A "Compositor" defines an ordered sequence of sub-elements (child elements). Examples of compositors are identified in Table C-2.

Table C-2 Compositors

Compositor	Diagram	Description
Sequence		In this example, a sequence of elements is used for defining an **Altova** element instance. These sequences of elements include a single Name instance, which is a simple type, and may contain an unlimited number of **Division** instances. **Name** must precede **Division**.
Choice		In this example a choice of elements are offered for defining an **Altova** element instance. These choices include a single **Name** instance, which is a simple type, or an unlmited number of **Division** instances.
All		In this example, the elements are used for defining an **Altova** element instance that may be in any order. These elements include a single **Name** instance, which is a simple type, and may contain an unlimited number of **Division** instances. It makes no difference if **Division** precedes **Name** or not.

This page left intentionally blank