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
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
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

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.

Page 5 of 118
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.
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

<table>
<thead>
<tr>
<th>Document Number</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>SISO-STD-003.0</td>
<td>Guide for Base Object Model (BOM) Use and Implementation</td>
</tr>
<tr>
<td>XML 1.0 (Second Edition)</td>
<td>Extensible Markup Language (XML) 1.0 (Second Edition) W3C Recommendation, 6 October 2000</td>
</tr>
</tbody>
</table>

Table 2-2 - Other Reference Documents

<table>
<thead>
<tr>
<th>Document Number</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE Std 1516.3-2003</td>
<td>IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP)</td>
</tr>
<tr>
<td>N/A</td>
<td>BOM PDG Product Nomination – March 2003</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

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

The definitions identified in 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

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstraction</td>
<td>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.</td>
</tr>
<tr>
<td>Base Object Model</td>
<td>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.</td>
</tr>
<tr>
<td>BOM Assembly</td>
<td>A composition of BOMs that can result in a Federation Object Model (FOM), Simulation Object Model (SOM), or pattern which encompasses a larger scope.</td>
</tr>
<tr>
<td>BOM Component Implementation</td>
<td>A component containing the functionality described by a BOM for a specific language or platform.</td>
</tr>
<tr>
<td>Component</td>
<td>“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&amp;S Metadata Focus Group</td>
</tr>
<tr>
<td>Composability</td>
<td>“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&amp;S Master Plan</td>
</tr>
<tr>
<td>Conceptual Entity</td>
<td>An abstract representation of a real world entity, phenomenon, process, or system. Hereafter referred to as entity.</td>
</tr>
<tr>
<td>Conceptual Event</td>
<td>A representation of a transient action that occurs among conceptual entities that may affect the state of one or more of the conceptual entities. Hereafter referred to as event.</td>
</tr>
<tr>
<td>Conceptual Model</td>
<td>A description of &quot;what the [simulation or federation] will represent, the assumptions limiting those representations, and other capabilities needed to satisfy the user's requirements.&quot;</td>
</tr>
<tr>
<td>Federate</td>
<td>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 &quot;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).&quot;</td>
</tr>
</tbody>
</table>

---


Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Federation</td>
<td>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</td>
</tr>
<tr>
<td>Glyph</td>
<td>An image used for the visual representation of a BOM such as within a tool palette or web page.</td>
</tr>
<tr>
<td>Message</td>
<td>An event that is directed to a specific receiver(s).</td>
</tr>
<tr>
<td>Metadata</td>
<td>“Structured, encoded data that describe characteristics of information-bearing entities to aid in the identification, discovery, assessment, and management of the described entities.”⁴</td>
</tr>
<tr>
<td>Pattern</td>
<td>A named set of recurring behavior used to accomplish a common objective, capability, or purpose.</td>
</tr>
<tr>
<td>Pattern Action</td>
<td>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.</td>
</tr>
<tr>
<td>Pattern of Interplay</td>
<td>Specific type of pattern characterized by a sequence of pattern actions involving one or more conceptual entities.</td>
</tr>
<tr>
<td>Simulation Space</td>
<td>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.</td>
</tr>
<tr>
<td>State Machine</td>
<td>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.</td>
</tr>
<tr>
<td>Trigger</td>
<td>An event that is not directed to a specific receiver(s).</td>
</tr>
</tbody>
</table>


4 Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOM</td>
<td>Base Object Model</td>
</tr>
<tr>
<td>BCI</td>
<td>BOM Component Implementation</td>
</tr>
<tr>
<td>BNF</td>
<td>Backus-Naur Form</td>
</tr>
<tr>
<td>COI</td>
<td>Community of Interest</td>
</tr>
<tr>
<td>DIF</td>
<td>Data Interchange Format</td>
</tr>
<tr>
<td>DoD</td>
<td>Department of Defense</td>
</tr>
<tr>
<td>EXCOM</td>
<td>Executive Committee</td>
</tr>
<tr>
<td>FDD</td>
<td>Federation Object Model (FOM) Document Data</td>
</tr>
<tr>
<td>FEDEP</td>
<td>Federation Development and Execution Process</td>
</tr>
<tr>
<td>FOM</td>
<td>Federation Object Model</td>
</tr>
<tr>
<td>HLA</td>
<td>High Level Architecture</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers, Inc.</td>
</tr>
<tr>
<td>M&amp;S</td>
<td>Modeling and Simulation</td>
</tr>
<tr>
<td>OMT</td>
<td>Object Model Template</td>
</tr>
<tr>
<td>PDG</td>
<td>Product Development Group</td>
</tr>
<tr>
<td>POC</td>
<td>Point of Contact</td>
</tr>
<tr>
<td>RFOM</td>
<td>Reference Federation Object Model</td>
</tr>
<tr>
<td>RTI</td>
<td>Runtime Infrastructure</td>
</tr>
<tr>
<td>SAC</td>
<td>Standards Activity Committee</td>
</tr>
<tr>
<td>SISO</td>
<td>Simulation Interoperability Standards Organization</td>
</tr>
<tr>
<td>SOM</td>
<td>Simulation Object Model</td>
</tr>
<tr>
<td>SRML</td>
<td>Simulation Reference Markup Language</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifiers</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>XSD</td>
<td>XML Schema Definition</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Category</th>
<th>Template Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Identification</td>
<td>Model Identification (Metadata)</td>
<td>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.</td>
</tr>
<tr>
<td>Conceptual Model Definition</td>
<td>Pattern of Interplay</td>
<td>Identifies the patterns of interplay including the pattern actions, variations, and exceptions required for representing the activities associated with each pattern of interplay.</td>
</tr>
<tr>
<td></td>
<td>State Machine</td>
<td>Identifies the state machines including the identification of conceptual entities and states required for representing the aspects of the conceptual model.</td>
</tr>
<tr>
<td></td>
<td>Entity Type</td>
<td>Identifies the types of conceptual entities required for representing the aspects of the conceptual model.</td>
</tr>
<tr>
<td></td>
<td>Event Type</td>
<td>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.</td>
</tr>
<tr>
<td>Model Mapping</td>
<td>Entity Type Mapping</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Event Type Mapping</td>
<td>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.</td>
</tr>
<tr>
<td>Object Model Definition</td>
<td>Object Class Structure</td>
<td>Records the list of HLA object classes related to patterns of interplay and describes their class-subclass relationships. This structure is unchanged from the HLA OMT Specification.</td>
</tr>
<tr>
<td></td>
<td>Interaction Class Structure</td>
<td>Records the list of HLA interaction classes related to patterns of interplay and describes their class-subclass relationships. This structure is unchanged from the HLA OMT Specification.</td>
</tr>
<tr>
<td></td>
<td>Attribute</td>
<td>Specifies features of HLA object class attributes in a pattern of interplay. This structure is unchanged from the HLA OMT Specification.</td>
</tr>
<tr>
<td></td>
<td>Parameter</td>
<td>Specifies features of HLA interaction class parameters in a pattern of interplay. This structure is unchanged from the HLA OMT Specification.</td>
</tr>
</tbody>
</table>
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>).

---

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>
<thead>
<tr>
<th>Category</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>&lt;name&gt;</td>
</tr>
<tr>
<td>Type</td>
<td>&lt;type&gt;</td>
</tr>
<tr>
<td>Version</td>
<td>&lt;version&gt;</td>
</tr>
<tr>
<td>Modification Date</td>
<td>&lt;date&gt;</td>
</tr>
<tr>
<td>Security Classification</td>
<td>&lt;security classification&gt;</td>
</tr>
<tr>
<td>Release Restriction</td>
<td>&lt;release restriction&gt;</td>
</tr>
<tr>
<td>Purpose</td>
<td>&lt;purpose&gt;</td>
</tr>
<tr>
<td>Application Domain</td>
<td>&lt;application domain&gt;</td>
</tr>
<tr>
<td>Description</td>
<td>&lt;description&gt;</td>
</tr>
<tr>
<td>Use Limitation</td>
<td>&lt;limitation&gt;</td>
</tr>
<tr>
<td>Use History</td>
<td>&lt;history&gt;</td>
</tr>
<tr>
<td>Keyword</td>
<td></td>
</tr>
<tr>
<td>Taxonomy</td>
<td>&lt;taxonomy&gt;</td>
</tr>
<tr>
<td>Keyword Value</td>
<td>&lt;keyword value&gt;</td>
</tr>
<tr>
<td>POC</td>
<td></td>
</tr>
<tr>
<td>POC Type</td>
<td>&lt;poc type&gt;</td>
</tr>
<tr>
<td>POC Name</td>
<td>&lt;poc name&gt;</td>
</tr>
<tr>
<td>POC Organization</td>
<td>&lt;poc organization&gt;</td>
</tr>
<tr>
<td>POC Telephone</td>
<td>&lt;poc telephone&gt;</td>
</tr>
<tr>
<td>POC Email</td>
<td>&lt;poc email&gt;</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Occurs</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>This field shall specify the name assigned to the object model.</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>Type</td>
<td>This field shall specify the type that the object model represents; for a BOM the only valid value is “BOM.”</td>
<td>1</td>
<td>BOM</td>
</tr>
<tr>
<td>Version</td>
<td>This field shall specify the version identification assigned to the object model.</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>Modification Date</td>
<td>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).</td>
<td>1</td>
<td>yyyy-mm-dd</td>
</tr>
<tr>
<td>Security Classification</td>
<td>This field shall contain the security classification of the object model.</td>
<td>1</td>
<td>Unclassified, Confidential, Secret, Top Secret, other text</td>
</tr>
<tr>
<td>Release Restriction</td>
<td>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.</td>
<td>0..many</td>
<td>text</td>
</tr>
<tr>
<td>Purpose</td>
<td>This field shall specify the purpose for which the BOM was developed.</td>
<td>0..1</td>
<td>text</td>
</tr>
<tr>
<td>Application Domain</td>
<td>This field shall specify the type or class of application to which the BOM pertains.</td>
<td>0..1</td>
<td>Analysis, Training, Test and Evaluation, Engineering, Acquisition, other text</td>
</tr>
<tr>
<td>Description</td>
<td>This field shall provide an account of the content of the BOM.</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>Use Limitation</td>
<td>This field shall provide any known applications for which this BOM has been found not to be appropriate.</td>
<td>0..1</td>
<td>text</td>
</tr>
<tr>
<td>Use History</td>
<td>This field shall provide a description of where this BOM has been used in the construction of other object models.</td>
<td>0..many</td>
<td>text</td>
</tr>
<tr>
<td>Keyword Taxonomy</td>
<td>This field specifies the source of the keyword vocabulary.</td>
<td>0..1</td>
<td>text</td>
</tr>
<tr>
<td>Keyword Value</td>
<td>This field shall provide the word or concept that is addressed by the BOM.</td>
<td>1</td>
<td>text</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Category</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Restaurant Payment</td>
</tr>
<tr>
<td>Type</td>
<td>BOM</td>
</tr>
<tr>
<td>Version</td>
<td>1.0 Beta</td>
</tr>
<tr>
<td>Modification Date</td>
<td>2004-05-04</td>
</tr>
<tr>
<td>Security Classification</td>
<td>Unclassified</td>
</tr>
<tr>
<td>Release Restriction</td>
<td>Not for release outside the XYZ Restaurant Corporation.</td>
</tr>
<tr>
<td>Release Restriction</td>
<td>Release only to BOM PUG members.</td>
</tr>
<tr>
<td>Purpose</td>
<td>Standardize the pattern of interplay between Customer and Waiter for payment of meal at restaurant.</td>
</tr>
<tr>
<td>Application Domain</td>
<td>Restaurant operations</td>
</tr>
<tr>
<td>Description</td>
<td>This is an example to illustrate the key concepts of a BOM.</td>
</tr>
<tr>
<td>Use Limitation</td>
<td>Not applicable to drive-through restaurants</td>
</tr>
<tr>
<td>Use History</td>
<td>Used as example in the BOM Template Specification</td>
</tr>
<tr>
<td>Keyword</td>
<td></td>
</tr>
<tr>
<td>Taxonomy</td>
<td>Restaurant taxonomy</td>
</tr>
<tr>
<td>Keyword Value</td>
<td>Restaurant</td>
</tr>
<tr>
<td>Keyword</td>
<td></td>
</tr>
<tr>
<td>Taxonomy</td>
<td>Commerce taxonomy</td>
</tr>
<tr>
<td>Keyword Value</td>
<td>Payment</td>
</tr>
<tr>
<td>POC</td>
<td></td>
</tr>
<tr>
<td>POC Type</td>
<td>Primary author</td>
</tr>
<tr>
<td>POC Name</td>
<td>Mr. Snuffy Smith</td>
</tr>
<tr>
<td>POC Organization</td>
<td>XYZ Restaurant Corporation</td>
</tr>
<tr>
<td>POC Telephone</td>
<td>+1 44 123-456-7890</td>
</tr>
<tr>
<td>POC Email</td>
<td><a href="mailto:snuffy.smith@goodeatscafe.com">snuffy.smith@goodeatscafe.com</a></td>
</tr>
<tr>
<td>POC</td>
<td></td>
</tr>
<tr>
<td>POC Type</td>
<td>Release authority</td>
</tr>
<tr>
<td>POC Name</td>
<td>Na</td>
</tr>
<tr>
<td>POC Organization</td>
<td>XYZ Restaurant Corporation</td>
</tr>
<tr>
<td>POC Telephone</td>
<td>+1 44 123-456-1000</td>
</tr>
<tr>
<td>POC Email</td>
<td><a href="mailto:the.lawyer@xyz-foods.com">the.lawyer@xyz-foods.com</a></td>
</tr>
<tr>
<td>Reference</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Glossary</td>
</tr>
<tr>
<td>Identification</td>
<td>ISBN 12345678901</td>
</tr>
<tr>
<td>Reference</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Conceptual Model</td>
</tr>
<tr>
<td>Identification</td>
<td><a href="http://boms.info/restaurantconceptualmodel.doc">http://boms.info/restaurantconceptualmodel.doc</a></td>
</tr>
<tr>
<td>Other</td>
<td>This BOM pattern of interplay was featured on the Food Network special “HLA and the Future of Restaurant Operations”</td>
</tr>
<tr>
<td>Glyph</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>JPG</td>
</tr>
<tr>
<td>Alt</td>
<td>Payment</td>
</tr>
<tr>
<td>Height</td>
<td>32</td>
</tr>
<tr>
<td>Width</td>
<td>32</td>
</tr>
<tr>
<td>Note</td>
<td><em>[1]</em></td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Pattern of Interplay Name</th>
<th>Sequence</th>
<th>Name</th>
<th>Sender</th>
<th>Receiver</th>
<th>Event</th>
<th>BOM</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern Action</td>
<td>&lt;sequence&gt;</td>
<td>&lt;pattern action name&gt;</td>
<td>&lt;sender name&gt;</td>
<td>&lt;receiver name&gt;</td>
<td>&lt;event name&gt;</td>
<td>&lt;BOM name&gt;</td>
<td></td>
</tr>
<tr>
<td>Exception</td>
<td>&lt;exception name&gt;</td>
<td>&lt;sender name&gt;</td>
<td>&lt;receiver name&gt;</td>
<td>&lt;event name&gt;</td>
<td>&lt;BOM name&gt;</td>
<td>&lt;condition&gt;</td>
<td></td>
</tr>
<tr>
<td>Variation</td>
<td>&lt;variation name&gt;</td>
<td>&lt;sender name&gt;</td>
<td>&lt;receiver name&gt;</td>
<td>&lt;event name&gt;</td>
<td>&lt;BOM name&gt;</td>
<td>&lt;condition&gt;</td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
parentage to uniquely identify the entity type (i.e., sender, receiver), event type (i.e., event) or bom type (i.e., BOM) being used.

![Diagram of BOM Pattern Action Relationship](image)

**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

<table>
<thead>
<tr>
<th>Pattern of Interplay Name</th>
<th>Seq</th>
<th>Name</th>
<th>Sender</th>
<th>Receiver</th>
<th>Event</th>
<th>BOM</th>
<th>Condition</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern Action 1</td>
<td>1</td>
<td>CustomerDone</td>
<td>CustomerEntity</td>
<td>Waiter</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>[2]</td>
</tr>
<tr>
<td>Variation</td>
<td></td>
<td>CustomerIdle</td>
<td>Table Entity</td>
<td>na</td>
<td>DirtyDishesOnTable</td>
<td>na</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td>Variation</td>
<td></td>
<td>CustomerRequest</td>
<td>CustomerEntity</td>
<td>Waiter</td>
<td>CheckRequest</td>
<td>na</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td>Pattern Action 2</td>
<td>2</td>
<td>CheckBroughtToTable</td>
<td>Waiter Entity</td>
<td>CustomerEntity</td>
<td>PaymentDueNotification</td>
<td>na</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td>Variation</td>
<td></td>
<td>CustomerPays</td>
<td>CustomerEntity</td>
<td>Waiter</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td>Variation</td>
<td></td>
<td>BillPaid_Cash</td>
<td>CustomerEntity</td>
<td>Waiter</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td>Variation</td>
<td></td>
<td>BillPaid_Credit</td>
<td>CustomerEntity</td>
<td>Waiter</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td>Exception</td>
<td></td>
<td>BillNotPaid</td>
<td>CustomerEntity</td>
<td>na</td>
<td>NonPaying</td>
<td>Customer</td>
<td>na</td>
<td>customer leaves without paying bill</td>
</tr>
<tr>
<td>Pattern Action 3</td>
<td>3</td>
<td>CustomerLeavesTip</td>
<td>CustomerEntity</td>
<td>Waiter</td>
<td>Tip</td>
<td>na</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td>Pattern Action 4</td>
<td>4</td>
<td>CustomerLeavesTip</td>
<td>CustomerEntity</td>
<td>Waiter</td>
<td>Tip</td>
<td>na</td>
<td>na</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

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

<table>
<thead>
<tr>
<th>State Machine Name</th>
<th>Conceptual Entities</th>
<th>State Name</th>
<th>Exit Condition</th>
<th>Exit</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;state machine&gt;</td>
<td>&lt;entity type name&gt;,&lt;entity type name&gt;</td>
<td>&lt;state name&gt;</td>
<td>&lt;exit action&gt;</td>
<td>&lt;next state&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;state machine&gt;</td>
<td>&lt;entity type name&gt;,&lt;entity type name&gt;</td>
<td>&lt;state name&gt;</td>
<td>&lt;exit action&gt;</td>
<td>&lt;next state&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;state machine&gt;</td>
<td>&lt;entity type name&gt;,&lt;entity type name&gt;</td>
<td>&lt;state name&gt;</td>
<td>&lt;exit action&gt;</td>
<td>&lt;next state&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;state machine&gt;</td>
<td>&lt;entity type name&gt;,&lt;entity type name&gt;</td>
<td>&lt;state name&gt;</td>
<td>&lt;exit action&gt;</td>
<td>&lt;next state&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
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**

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Occurs</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Machine</td>
<td>Identifies a state machine.</td>
<td>1..many</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Identifies the name of the state machine.</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>Conceptual Entities</td>
<td>Identifies one or more conceptual entity types that can represent the states defined.</td>
<td>1..many</td>
<td>entity/identifier</td>
</tr>
<tr>
<td>State</td>
<td>Identifies a behavior state which a conceptual entity may exhibit.</td>
<td>1..many</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Identifies the name of the state to be described.</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>Exit Condition</td>
<td>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.</td>
<td>0..many</td>
<td></td>
</tr>
<tr>
<td>Exit Action</td>
<td>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.</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>Next State</td>
<td>Identifies which state succeeds the present state based on the condition of the exit action.</td>
<td>1</td>
<td>text</td>
</tr>
</tbody>
</table>

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

![Figure 6-5 BOM State Machine Relationship](image)

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**

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

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.

Page 33 of 118
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

<table>
<thead>
<tr>
<th>Entity Type Name</th>
<th>Entity Type Characteristic Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;entity type name&gt;</td>
<td>&lt;characteristic name&gt;, &lt;characteristic name&gt;</td>
</tr>
<tr>
<td>&lt;entity type name&gt;</td>
<td>&lt;characteristic name&gt;, &lt;characteristic name&gt;</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Occurs</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type</td>
<td>Identifies an entity type within the conceptual model</td>
<td>1..many</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Identifies a unique name for the BOM entity type</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>Characteristic</td>
<td>Identifies an attribute-like quality associated with the entity type</td>
<td>1..many</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Identifies a unique name for the characteristic</td>
<td>1</td>
<td>text</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Entity Type Name</th>
<th>Entity Type Characteristic Name</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>TableEntity</td>
<td>ID</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>Occupied</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>Dishes_Dirty</td>
<td>na</td>
</tr>
<tr>
<td>CustomerEntity</td>
<td>ID</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>Table_ID</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>FoodOrder</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>PaidBill</td>
<td>na</td>
</tr>
<tr>
<td>WaiterEntity</td>
<td>ID</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>Assigned_Tables</td>
<td>na</td>
</tr>
<tr>
<td>BillEntity</td>
<td>Amount</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>Tip</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>Waiter_ID</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>Table_ID</td>
<td>na</td>
</tr>
<tr>
<td>Note</td>
<td>na</td>
<td></td>
</tr>
</tbody>
</table>

### 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.
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](image)

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

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
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

<table>
<thead>
<tr>
<th>Event Type Name</th>
<th>Source Characteristic Name</th>
<th>Target Characteristic Name</th>
<th>Content Characteristic Name</th>
<th>Trigger Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;event type&gt;</td>
<td>&lt;source char&gt;</td>
<td>&lt;target char&gt;,</td>
<td>&lt;content char&gt;,</td>
<td>&lt;trigger expression&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;target char&gt;,</td>
<td>&lt;content char&gt;,</td>
<td>&lt;trigger expression&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;target char&gt;</td>
<td>&lt;content char&gt;,</td>
<td>&lt;trigger expression&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;target char&gt;,</td>
<td>&lt;content char&gt;,</td>
<td>&lt;trigger expression&gt;</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Occurs</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Type</td>
<td>Identifies an event type within the conceptual model.</td>
<td>0..many</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Identifies a unique name for the BOM event type.</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>Source Character</td>
<td>Identifies a characteristic defined within the Entity Type Table to be associated as the source of the event.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>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.</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>Target Character</td>
<td>Identifies a characteristic defined within the Entity Type Table to be associated as a target of the event.</td>
<td>0..many</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>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.</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>Content Character</td>
<td>Identifies a characteristic defined within the Entity Type Table to be associated as a content of the event.</td>
<td>0..many</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>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.</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>Trigger Condition</td>
<td>Specifies the condition in the form of a Boolean expression associated with triggering the event type.</td>
<td>0..1</td>
<td>text</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Event Type Name</th>
<th>Source Characteristic Name</th>
<th>Target Characteristic Name</th>
<th>Content Characteristic Name</th>
<th>Event Type Name</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirtyDishesOnTable</td>
<td>TableEntity.ID</td>
<td>na</td>
<td>na</td>
<td>TableEntity.Dishes_Dirty = TRUE</td>
<td>na</td>
</tr>
<tr>
<td>CheckRequest</td>
<td>CustomerEntity.ID</td>
<td>WaiterEntity.ID</td>
<td>na</td>
<td>na</td>
<td>[4]</td>
</tr>
<tr>
<td>PaymentDueNotification</td>
<td>WaiterEntity.ID</td>
<td>CustomerEntity.ID</td>
<td>BillEntity.Amount</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td>Tip</td>
<td>CustomerEntity.ID</td>
<td>WaiterEntity.ID</td>
<td>BillEntity.Tip</td>
<td>na</td>
<td></td>
</tr>
</tbody>
</table>

#### 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>
<thead>
<tr>
<th>Table 6-17 Entity Type Mapping Table Format</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entity Type Mapping</strong></td>
</tr>
<tr>
<td>Entity Type Name</td>
</tr>
<tr>
<td>&lt;entity type&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>&lt;entity type&gt;</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Table 6-18 Entity Type Mapping Information Description Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Category</strong></td>
</tr>
<tr>
<td>Entity Type Mapping</td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>HLA Object/Interaction Class</td>
</tr>
<tr>
<td>Characteristic</td>
</tr>
<tr>
<td>HLA Attribute/Parameter</td>
</tr>
</tbody>
</table>

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

![BOM Entity Type Mapping Relationship](image-url)

**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>
<thead>
<tr>
<th>Entity Type Mapping</th>
<th>Entity Characteristic Mapping</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type Name</td>
<td>HLA Object/Interaction Class</td>
<td></td>
</tr>
<tr>
<td>TableEntity</td>
<td>HLAObjectRoot.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PhysicalEntity.Table</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ID</td>
<td>Table.ID</td>
</tr>
<tr>
<td></td>
<td>Occupied</td>
<td>Table.isOccupied</td>
</tr>
<tr>
<td></td>
<td>Dishes_Dirty</td>
<td>Table.hasDirtyDishes</td>
</tr>
<tr>
<td>CustomerEntity</td>
<td>HLAObjectRoot.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PhysicalEntity.Human.Customer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ID</td>
<td>Customer.ID</td>
</tr>
<tr>
<td></td>
<td>Table_ID</td>
<td>Customer.AssignedTableID</td>
</tr>
<tr>
<td></td>
<td>FoodOrder</td>
<td>Customer.Order</td>
</tr>
<tr>
<td></td>
<td>PaidBill</td>
<td>Customer.hasPaid</td>
</tr>
<tr>
<td>WaiterEntity</td>
<td>HLAObjectRoot.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PhysicalEntity.Waiter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ID</td>
<td>Waiter.ID</td>
</tr>
<tr>
<td></td>
<td>Assigned_Tables</td>
<td>Waiter.AssignedTables[]</td>
</tr>
<tr>
<td>BillEntity</td>
<td>HLAObjectRoot.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PhysicalEntity.Payment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Amount</td>
<td>Payment.Amount</td>
</tr>
<tr>
<td></td>
<td>Tip</td>
<td>Payment.Tip</td>
</tr>
<tr>
<td></td>
<td>HLAObjectRoot.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PhysicalEntity.Waiter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Waiter_ID</td>
<td>Waiter.ID</td>
</tr>
<tr>
<td></td>
<td>HLAObjectRoot.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PhysicalEntity.Human.Customer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Table_ID</td>
<td>Customer.AssignedTableID</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Event Type Mapping</th>
<th>Source Characteristic Mapping</th>
<th>Target Characteristic Mapping</th>
<th>Content Characteristic Mapping</th>
<th>Trigger Condition Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>HLA Object/Interaction Classes</td>
<td>Name</td>
<td>HLA Attrs/Params</td>
<td>Name</td>
</tr>
<tr>
<td>&lt;event type&gt;</td>
<td>&lt;HLA class&gt;</td>
<td>&lt;source char&gt;</td>
<td>&lt;HLA attr/param&gt;</td>
<td>&lt;target char&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;event type&gt;</td>
<td>&lt;HLA class&gt;</td>
<td></td>
<td></td>
<td>&lt;content char&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source char&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;event type&gt;</td>
<td>&lt;HLA class&gt;</td>
<td></td>
<td></td>
<td>&lt;target char&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Occurs</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Type Mapping</td>
<td>Identifies an event type mapping.</td>
<td>1..many</td>
<td></td>
</tr>
<tr>
<td>Checks</td>
<td>References an event type defined in the Event Type Definition Table.</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>HLA Object/Interaction Class</td>
<td>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.</td>
<td>1..many</td>
<td>text</td>
</tr>
<tr>
<td>Source Characteristic</td>
<td>Identifies a source characteristic associated with the event type identified in the Event Type Definition Table to be mapped.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Notes Table</td>
<td>References a specific characteristic associated with the event type.</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>HLA Attribute/Parameter</td>
<td>References an HLA object class attribute or HLA interaction class parameter that can be used to represent the specified event type. 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.</td>
<td>1..many</td>
<td>text</td>
</tr>
<tr>
<td>Target Characteristic</td>
<td>Identifies a characteristic defined within the Entity Type Table to be associated as a target of the event type.</td>
<td>0..many</td>
<td></td>
</tr>
<tr>
<td>Notes Table</td>
<td>References a specific characteristic associated with the event type.</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>HLA Attribute/Parameter</td>
<td>References an HLA object class attribute or HLA interaction class parameter that can be used to represent the specified event type. 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.</td>
<td>1..many</td>
<td>text</td>
</tr>
<tr>
<td>Content Characteristic</td>
<td>Identifies a characteristic defined within the Entity Type Table to be associated as a content of the event type.</td>
<td>0..many</td>
<td></td>
</tr>
<tr>
<td>Notes Table</td>
<td>References a specific characteristic associated with the event type.</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>HLA Attribute/Parameter</td>
<td>References an HLA object class attribute or HLA interaction class parameter that can be used to represent the specified event type. 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.</td>
<td>1..many</td>
<td>text</td>
</tr>
<tr>
<td>Trigger Condition</td>
<td>Specifies the condition in the form of a Boolean expression associated with triggering the event type.</td>
<td>0..1</td>
<td></td>
</tr>
<tr>
<td>Notes Table</td>
<td>References a specific trigger condition expression associated with the event type.</td>
<td>1</td>
<td>text</td>
</tr>
</tbody>
</table>

---

**Permission:** This is an approved SISO Standard.
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

<table>
<thead>
<tr>
<th>Event Type</th>
<th>HLA Object / Interaction Classes</th>
<th>Source Characteristic</th>
<th>Target Characteristic</th>
<th>Content Characteristic</th>
<th>Trigger Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirtyDishesOnTable</td>
<td>HLAObjectRoot. PhysicalEntity.Table</td>
<td>TableEntity.ID</td>
<td>Table.ID</td>
<td>na</td>
<td>TableEntity. DishesDirty = TRUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Table. hasDirtyDishes = TRUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Customer. hasPaid = FALSE</td>
</tr>
<tr>
<td>CheckRequest</td>
<td>RealWorldMessage. RequestBill</td>
<td>CustomerEntity.ID</td>
<td>RequestBill. CustomerID</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>PaymentDueNotification</td>
<td>RealWorldMessage. PaymentDue</td>
<td>WaiterEntity.ID</td>
<td>PaymentDue. WaiterID</td>
<td>PaymentDue. CustomerID</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PaymentDueAmount. Amount</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>na</td>
</tr>
</tbody>
</table>

**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](image)

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

<table>
<thead>
<tr>
<th>Object Class Definition</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLAObjectRoot (N)</td>
<td></td>
</tr>
<tr>
<td>PhysicalEntity</td>
<td>Customer (PS)</td>
</tr>
<tr>
<td></td>
<td>Waiter (PS)</td>
</tr>
<tr>
<td>AbstractEntity</td>
<td>Table (PS)</td>
</tr>
<tr>
<td></td>
<td>Payment (PS)</td>
</tr>
</tbody>
</table>

Table 6-24 HLA Attributes Example

<table>
<thead>
<tr>
<th>Object Class Ref</th>
<th>Attribute</th>
<th>Datatype</th>
<th>Update Type</th>
<th>Update Condition</th>
<th>D/A</th>
<th>P/S</th>
<th>Availble Dimensions</th>
<th>Transp ortatio n</th>
<th>Order</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLAObjectRoot</td>
<td>HLApriilege</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>PhysicalEntity</td>
<td>ID</td>
<td>ID</td>
<td>Static</td>
<td></td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>PhysicalEntity.</td>
<td>hasDirtyDishes</td>
<td>Boolean</td>
<td>Conditional</td>
<td>on change or request</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>Table</td>
<td>isOccupied</td>
<td>Boolean</td>
<td>Conditional</td>
<td>on change or request</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>PhysicalEntity.</td>
<td>AssignedTables</td>
<td>Assigned Tables Type</td>
<td>Conditional</td>
<td>on change or request</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>Customer</td>
<td>TableID</td>
<td>ID</td>
<td>Static</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>Order</td>
<td>OrderType</td>
<td>Conditional</td>
<td>on change or request</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>hasPaid</td>
<td>Boolean</td>
<td>Conditional</td>
<td>on change or request</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>AbstractEntity.</td>
<td>Amount</td>
<td>Float32BE</td>
<td>Conditional</td>
<td>on change or request</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>Payment</td>
<td>Tip</td>
<td>Float32BE</td>
<td>Conditional</td>
<td>on change or request</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>Source</td>
<td>ID</td>
<td>Static</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>Recipient</td>
<td>ID</td>
<td>Static</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Interaction Class Definition</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLAInteractionRoot (N)</td>
<td></td>
</tr>
<tr>
<td>Physical Event</td>
<td>CustomerLeaves</td>
</tr>
<tr>
<td>RealWorldMessage</td>
<td>RequestBill</td>
</tr>
<tr>
<td>PaymentDue</td>
<td>na</td>
</tr>
</tbody>
</table>

Table 6-26 HLA Parameters Example

<table>
<thead>
<tr>
<th>Interaction Class Ref</th>
<th>Parameter</th>
<th>Datatype</th>
<th>Available Dimensions</th>
<th>Transportation</th>
<th>Order</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLAInteractionRoot. PhysicalEvent</td>
<td>CustomerID</td>
<td>ID</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>HLAInteractionRoot. RealWorldMessage</td>
<td>CustomerID</td>
<td>ID</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>RealWorldMessage. PaymentDue</td>
<td>WaiterID</td>
<td>ID</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>RealWorldMessage. RequestBill</td>
<td>WaiterID</td>
<td>ID</td>
<td>na</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>Note</td>
<td></td>
<td>na</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

### 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.
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

<table>
<thead>
<tr>
<th>Pattern of Interplay</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RestaurantPayment</td>
<td>Payment of a meal at a restaurant</td>
</tr>
<tr>
<td>RestaurantVisitation</td>
<td>Customer visitation activities at a restaurant</td>
</tr>
<tr>
<td>FoodOrder</td>
<td>Actions associated with completing a food order</td>
</tr>
<tr>
<td>AbstractEntity.CashPayment</td>
<td>Represents a payment using cash</td>
</tr>
</tbody>
</table>

#### 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

<table>
<thead>
<tr>
<th>Pattern of Interplay</th>
<th>Pattern Action</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;pattern of interplay&gt;</td>
<td>&lt;action&gt;</td>
<td>&lt;definition&gt;</td>
</tr>
</tbody>
</table>

- 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

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

<table>
<thead>
<tr>
<th>State Machine</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;state machine&gt;</td>
<td>&lt;definition&gt;</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>&lt;state machine&gt;</td>
<td>&lt;definition&gt;</td>
</tr>
</tbody>
</table>

- 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

<table>
<thead>
<tr>
<th>State Machine</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmployeeSM</td>
<td>Employee of a restaurant</td>
</tr>
<tr>
<td>CustomerSM</td>
<td>Common customer actions</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>State Machine</th>
<th>State</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;state machine&gt;</td>
<td>&lt;state&gt;</td>
<td>&lt;definition&gt;</td>
</tr>
<tr>
<td>&lt;state machine&gt;</td>
<td>&lt;state&gt;</td>
<td>&lt;definition&gt;</td>
</tr>
</tbody>
</table>

- 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

<table>
<thead>
<tr>
<th>State Machine</th>
<th>State</th>
<th>Definition</th>
</tr>
</thead>
</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>
<thead>
<tr>
<th>Entity Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmployeeSM Ready</td>
<td>Employee is ready to serve</td>
</tr>
<tr>
<td>EmployeeSM Greet</td>
<td>Employee greets customer</td>
</tr>
<tr>
<td>EmployeeSM ProcessBill</td>
<td>Employee processes bill</td>
</tr>
<tr>
<td>EmployeeSM PrepareBill</td>
<td>Employee prepares bill</td>
</tr>
<tr>
<td>EmployeeSM ProcessOrder</td>
<td>Employee processes food order</td>
</tr>
<tr>
<td>EmployeeSM ClearingTable</td>
<td>Employee clears table</td>
</tr>
</tbody>
</table>

- 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>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>TableEntity</td>
<td>Place where meal is served</td>
</tr>
<tr>
<td>CustomerEntity</td>
<td>Person paying for a meal</td>
</tr>
<tr>
<td>WaiterEntity</td>
<td>Waiter who served the meal and processes payment</td>
</tr>
<tr>
<td>BillEntity</td>
<td>Money owed for meal</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Entity Type</th>
<th>Characteristic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;entity type&gt;</td>
<td>&lt;char&gt;</td>
<td>&lt;definition&gt;</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>&lt;entity type&gt;</td>
<td>&lt;char&gt;</td>
<td>&lt;definition&gt;</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Type</th>
<th>Characteristic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>TableEntity</td>
<td>Dishes_Dirty</td>
<td>That fact that the table has dirty dishes</td>
</tr>
<tr>
<td>CustomerEntity</td>
<td>ID</td>
<td>A Customer identifier</td>
</tr>
<tr>
<td>WaiterEntity</td>
<td>ID</td>
<td>A Waiter identifier</td>
</tr>
<tr>
<td></td>
<td>Assigned_Tables</td>
<td>List of tables for which Waiter is responsible</td>
</tr>
<tr>
<td>BillEntity</td>
<td>Amount</td>
<td>Amount of money</td>
</tr>
<tr>
<td></td>
<td>Waiter_ID</td>
<td>Identifies Waiter processing payment</td>
</tr>
<tr>
<td></td>
<td>Table_ID</td>
<td>Identifies Table at which the customer has been seated</td>
</tr>
</tbody>
</table>

### 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>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;event type&gt;</td>
<td>&lt;definition&gt;</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirtyDishesOnTable</td>
<td>There are dirty dishes on a table</td>
</tr>
<tr>
<td>NonPayingCustomer</td>
<td>Customer leaves without paying</td>
</tr>
<tr>
<td>CheckRequest</td>
<td>Request check from waiter</td>
</tr>
<tr>
<td>PaymentDueNotification</td>
<td>Notification of payment due</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>Type</th>
<th>Characteristic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;event type&gt;</td>
<td>&lt;char&gt;</td>
<td>&lt;definition&gt;</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>&lt;event type&gt;</td>
<td>&lt;char&gt;</td>
<td>&lt;definition&gt;</td>
</tr>
</tbody>
</table>

- 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 characteristics* are taken from the examples provided earlier in Section 6.2.4.

#### Table 6-44 BOM Event Type Characteristic Definition Table Example

<table>
<thead>
<tr>
<th>Type</th>
<th>Characteristic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirtyDishesOnTable</td>
<td>TableEntity.ID</td>
<td>Identifies Table</td>
</tr>
<tr>
<td></td>
<td>TableEntity.Dishes_Dirty</td>
<td>Table has dirty dishes</td>
</tr>
<tr>
<td>NonPayingCustomer</td>
<td>CustomerEntity.ID</td>
<td>Identifies Customer</td>
</tr>
<tr>
<td></td>
<td>CustomerEntity.hasPaid</td>
<td>Customer did not pay</td>
</tr>
<tr>
<td>CheckRequest</td>
<td>CustomerEntity.ID</td>
<td>Identifies Customer making the request</td>
</tr>
<tr>
<td></td>
<td>WaiterEntity.ID</td>
<td>Identifies Waiter to whom the request is directed</td>
</tr>
<tr>
<td>PaymentDueNotification</td>
<td>CustomerEntity.ID</td>
<td>Identifies Customer owing payment</td>
</tr>
<tr>
<td></td>
<td>Amount</td>
<td>The total amount</td>
</tr>
</tbody>
</table>
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](image_url)
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](image)

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.
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.
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](image)

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.
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.
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 entityType 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.
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.
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](image)

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

![Figure 7-16 BOM entityCharacteristicMappingsType Elements](image)

#### 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 \textit{sourceCharacteristic}, \textit{targetCharacteristic}, \textit{contentCharacteristic}, and \textit{triggerCondition} elements identified as part of the \textit{eventTypeMappingType} are represented using the complex type depicted in Figure 7-18.

![Figure 7-17 BOM eventTypeMappingsType Elements](image1)

![Figure 7-18 BOM eventCharacteristicMappingType Elements](image2)

### 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 \textit{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 \textit{objectModelDefinition} element in the BOM DIF Schema (see Figure 7.1) and are defined by the \textit{objectModelDefinitionType}. The elements used to represent the \textit{objectModelDefinitionType} of a BOM are depicted in Figure 7-19.
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](image-url)
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
<?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 -->


```

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
This is an approved SISO Standard.
<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:complexType>
<xs:attributeGroup ref="modelID:commonAttributes"/>
<xs:complexType name="stateType">
<xs:sequence>
<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:complexType>
</xs:element>
<xs:element name="semantics" type="modelID:String" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
<xs:attributeGroup ref="modelID:commonAttributes"/>
<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:sequence>
</xs:complexType>
<xs:attributeGroup ref="modelID:commonAttributes"/>
<xs:complexType name="objectModelDefinitionType">
<xs:all>
<xs:element name="objects" type="omt:objectsType" minOccurs="0" maxOccurs="0"/>
<xs:element name="interactions" type="omt:interactionsType" minOccurs="0" maxOccurs="0"/>
<xs:element name="dataTypes" type="omt:dataTypesType" minOccurs="0" maxOccurs="0"/>
</xs:all>
</xs:complexType>
<xs:attributeGroup ref="modelID:commonAttributes"/>
<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:sequence>
</xs:complexType>
<xs:attributeGroup ref="modelID:commonAttributes"/>
<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:complexType>
<xs:attributeGroup ref="modelID:commonAttributes"/>
<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:complexType>
<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.

Page 75 of 118
<xs:complexType name="eventTypeMappingType">
  <xs:sequence>
    <xs:element name="eventTypeMapping" type="eventTypeMappingType" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

SISO-STD-003-2006, BOM Template Specification

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
<xs:complexType>
  <xs:sequence>
    <xs:element name="entityTypeMappings" type="entityTypeMappingsType" minOccurs="0"/>
    <xs:element name="eventTypeMappings" type="eventTypeMappingsType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="conceptualModelType">
  <xs:sequence>
    <xs:element name="patternInterplay" type="patternInterplayType" id="patternID" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="entityTypes" minOccurs="0" maxOccurs="unbounded"/>
      <xs:complexType>
        <xs:sequence>
          <xs:element name="entityType" type="entityType" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    <xs:element name="eventTypes" minOccurs="0" maxOccurs="unbounded"/>
      <xs:complexType>
        <xs:sequence>
          <xs:element name="eventType" type="eventType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
  </xs:sequence>
</xs:complexType>
</xs:schema>
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
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

targetNamespace="http://www.sisostd.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:complexType name="OMTypeEnumerations">
    <xs:restriction base="xs:string">
      <xs:enumeration value="FOM"/>
      <xs:enumeration value="SOM"/>
      <xs:enumeration value="BOM"/>
    </xs:restriction>
  </xs:complexType>
  <xs:complexType name="OMTypeUnion">
    <xs:union memberTypes="OMTypeEnumerations nonEmptyString"/>
  </xs:complexType>
  <xs:complexType 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:complexType>
  <xs:complexType name="glyphTypeUnion">
    <xs:union memberTypes="glyphTypeEnumerations"/>
  </xs:complexType>
  <xs:complexType name="glyphType mixed" 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:schema>
```

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
<xs:complexType name="pocType">
    <xs:sequence>
        <xs:element name="pocName" type="xs:string" minOccurs="0"/>
        <xs:element name="pocOrg" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="pocTelephone" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="pocEmail" type="xs: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: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:simpleType name="modelType">
    <xs:simpleContent>
        <xs:extension base="OMTy">
            <xs:attributeGroup ref="commonAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:simpleType>

<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:complexType>
  <xs:sequence>
    <xs:element name="securityClassification" type="securityClassificationType">  
      <xs:annotation>
        <xs:documentation>specifies the security classification of the model</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:annotation>
        <xs:documentation>specifies keywords that characterize the model</xs:documentation>
      </xs:annotation>
    </xs:element>
    <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 characterizes the model</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="poc" maxOccurs="unbounded">  
      <xs:annotation>
        <xs:documentation>specify an organization or an whose has a particular role with respect to the model</xs:documentation>
      </xs:annotation>
    </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:element>
  </xs:sequence>
</xs:complexType>
<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
<?xml version="1.0" encoding="UTF-8"?>
  <xs:element name="objectModel" type="objectModelType">
    <xs:key name="dimensionDatatypeKey">
      <xs:field xpath="*:dimensionDatatypeKey"/>
    </xs:key>
    <xs:annotation>
      <xs:documentation>unique keys for identifying basic data representations</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="interactionClass">
    <xs:complexType>
      <xs:complexContent>
        <xs:restriction base="interactionClassType">
          <xs:attribute name="reference" type="string"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="objectClass">
    <xs:complexType>
      <xs:complexContent>
        <xs:restriction base="objectClassType">
          <xs:attribute name="reference" type="string"/>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
SISO-STD-003-2006, BOM Template Specification

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
<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:element any namespace="##other" minOccurs="0">
  <xs:任何 annotation>
    <xs:documentation>lexicon entry for this object class</xs:documentation>
  </xs:annotation>
</xs:element>

<xs:complexType name="objectClassType">
  <xs:sequence>
    <xs:element name="name" type="modelID:IdentifierType" />
    <xs:element name="sharing" type="SharingType" default="Neither" />
    <xs:documentation>specifies publication and subscription capabilities of this object</xs:documentation>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="interactionClassType">
  <xs:sequence>
    <xs:element name="name" type="modelID:IdentifierType" />
    <xs:element name="sharing" type="SharingType" />
    <xs:documentation>specifies publication and subscription capabilities of this interaction</xs:documentation>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="modelID:commonAttributes">
  <xs:sequence>
    <xs:element name="dimensions" minOccurs="0">
      <xs:annotation>
        <xs:documentation>records the association of the interaction class with a set of if a federate or federation is using DDM services for this attribute</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="transportation" 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:element any namespace="##other" minOccurs="0">
      <xs:任何 annotation>
        <xs:documentation>specifies the type of transportation used with this interaction</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</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:documentation>specifies publication and subscription capabilities of this object</xs:documentation>
  </xs:sequence>
</xs:complexType>

Copyright © 2006 SISO.  All rights reserved.
This is an approved SISO Standard.
<xs:element name="parameter" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="timeType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="interactionClass" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="modelID:commonAttributes" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:any namespace="#other" minOccurs="0"/>
  </xs:complexType>
</xs:element>
</xs:complexType>

<xs:complexType name="dimensionsType">
  <xs:sequence>
    <xs:element name="dimension" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:extension base="dimensionType">
          <xs:attributeGroup ref="modelID:commonAttributes"/>
        </xs:extension>
      </xs:complexType>
    </xs:element>
    <xs:any namespace="#other" minOccurs="0"/>
  </xs:sequence>
  <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:sequence>
    <xs:element name="upperBound" minOccurs="0">
      <xs:complexType>
        <xs:simpleContent>
          <xs:attributeGroup ref="modelID:commonAttributes"/>
          <xs:extension base="positiveInteger"/>
        </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's coordinates 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:element ref="#other" minOccurs="0"/>
    <xs:sequence>
      <xs:element ref="modelID:commonAttributes"/>
    </xs:sequence>
    <xs:complexType name="timeType">
      <xs:sequence>
        <xs:element name="timeStamp" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="dataType" type="ReferenceType"/>
            </xs:sequence>
            <xs:annotation>
              <xs:documentation>identifies the timestamp datatype</xs:documentation>
            </xs:annotation>
          </xs:complexType>
        </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:element ref="#other" minOccurs="0"/>
        <xs:sequence>
          <xs:attributeGroup ref="modelID:commonAttributes"/>
        </xs:sequence>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>

This is an approved SISO Standard.
SISO-STD-003-2006, BOM Template Specification

```xml
<xs:element name="lookahead" minOccurs="0">
  <xs:complexType>
    <xs:annotation>
      <xs:documentation>identifies the lookahead</xs:documentation>
    </xs:annotation>
    <xs:documentation> datatype</xs:documentation>
  </xs:complexType>
</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:complexType>
    <xs:annotation>
      <xs:documentation> indicates the level of interaction that a federate is capable of honoring</xs:documentation>
    </xs:annotation>
    <xs:element name="label" type="modelID:IdentifierType"/>
    <xs:element name="dataType" type="ReferenceType" minOccurs="0"/>
    <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:sequence>
    <xs:complexType name="tagsType">
      <xs:extension base="modelID:commonAttributes"/>
    </xs:complexType>
  </xs:complexType>
</xs:element>

<xs:element name="synchronizationPoint" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
    <xs:annotation>
      <xs:documentation> expands and describes the use of the synchronization point</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="synchronizationPoint" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:complexType name="transportationsType">
      <xs:annotation>
        <xs:documentation>designates as providing a user-supplied tag for those synchronization that the federate or federation designate as providing a user-supplied tag</xs:documentation>
      </xs:annotation>
    </xs:complexType>
  </xs:complexType>
</xs:element>
```

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
<xs:complexType>
  <xs:complexContent base="transportationType"/>
</xs:complexType>
<xs:complexType>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
  <xs:complexType name="transportationType">
    <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:complexType>
</xs:complexType>
<xs:complexType>
  <xs:complexType name="switchesType">
    <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:complexType>
</xs:complexType>
<xs:complexType>
  <xs:complexType name="switchesType">
    <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:complexType>
</xs:complexType>
<xs:complexType>
  <xs:complexType name="basicDataType">
    <xs:element name="interpretation" type="modelID:String"/>
    <xs:element name="endian" type="endianType"/>
    <xs:element name="encoding" type="modelID:String"/>
    <xs:any namespace="##other" minOccurs="0"/>
  </xs:complexType>
</xs:complexType>
<xs:complexType>
  <xs:complexType name="simpleDataType">
    <xs:element name="name" type="modelID:IdentifierType"/>
    <xs:element name="representation" type="ReferenceType"/>
    <xs:element name="units" type="modelID:NonEmptyString" default="NA" minOccurs="0"/>
    <xs:any namespace="##other" minOccurs="0"/>
  </xs:complexType>
</xs:complexType>

This is an approved SISO Standard.
enumerator fields) as delivered to and received from the RTI federation.

This is an approved SISO Standard.

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
<xs:annotation>
  <xs:documentation>identifies the datatype of an element of this array</xs:documentation>
</xs:annotation>
<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 sequence elements and the order of elements in multi-dimensional arrays) 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:element>
</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:sequence>
  <xs:documentation>identifies the datatype of the discriminant as delivered to and received from the RTI</xs:documentation>
</xs:complexType>
<xs:complexType name="variantRecordEncodingType">
  <xs:sequence>
    <xs:element name="name" type="modelID:IdentifierType" minOccurs="0"/>
    <xs:element name="dataType" type="ReferenceType" minOccurs="0"/>
  </xs:sequence>
  <xs:documentation>describe, in detail, the encoding of the array datatype (e.g., the location of the discriminant) as delivered to and received from the RTI</xs:documentation>
</xs:complexType>
<xs:element name="alternative" minOccurs="0" maxOccurs="unbounded">
  <xs:sequence>
    <xs:element name="enumerator" type="modelID:NonEmptyString">
      <xs:annotation>
        <xs:documentation>enumerators or enumerator ranges that determine the alternative</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:element>
<xs:element name="identification" type="modelID:IdentifierType" minOccurs="0">
  <xs:any namespace="#other" minOccurs="0"/>
</xs:element>
<x:s:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType name="notesType">
  <xs:sequence>
    <xs:element name="note" minOccurs="0" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="label" type="String"/>
        <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:annotation>
<xs:element name="semantics" type="modelID:String" minOccurs="0">
  <xs:any namespace="#other" minOccurs="0"/>
</xs:element>
<x:s:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:element name="any namespace="#other" minOccurs="0"/>
it is not used as a datatype, it forms the basis of the datatypes. Although
data items.

The variant record datatypes describes discriminated unions of types; also known as arrays or sequences.

The array datatypes describes indexed homogenous collections of datatypes; also known as records or structures.

The fixed record datatypes describes heterogeneous collections of types; also known as variant or choice records.

Basic data representation is the underpinning of all OMT datatypes. Although

data describes simple, scalar data values.

The enumerated datatypes describes data elements that can take on a finite discrete set of values.

The simple datatypes describes simple, scalar data.

The array datatypes describes indexed homogenous collections of datatypes; also known as records or structures.

The fixed record datatypes describes heterogeneous collections of types; also known as variant or choice records.
<xs:complexType name="arrayDataTypesType">
  <xs:sequence>
    <xs:element name="arrayData" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:complexContent>
          <xs:restriction base="arrayDataType">
            <xs:all/>
          </xs:restriction>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="fixedRecordDataTypesType">
  <xs:sequence>
    <xs:element name="fixedRecordData" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:complexContent>
          <xs:restriction base="fixedRecordDataType">
            <xs:all/>
          </xs:restriction>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="variantRecordDataTypesType">
  <xs:sequence>
    <xs:element name="variantRecordData" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:complexContent>
          <xs:restriction base="variantRecordDataType">
            <xs:all/>
          </xs:restriction>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</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</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</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:simpleContent>
  <xs:extension base="EndianEnumerations">
    <xs:attributeGroup ref="ModelID:CommonAttributes"/>
  </xs:extension>
</xs:simpleContent>

<xs:complexType name="OwnershipType">
  <xs:extension base="OwnershipEnumerations">
    <xs:attributeGroup ref="ModelID:CommonAttributes"/>
  </xs:extension>
</xs:complexType>

<xs:complexType name="CapabilityType">
  <xs:extension base="CapabilityEnumerations">
    <xs:attributeGroup ref="ModelID:CommonAttributes"/>
  </xs:extension>
</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:restriction base="xs:string">
      <xs:enumeration value="Receive"/>
      <xs:enumeration value="TimeStamp"/>
    </xs:restriction>
  </xs:annotation>
</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:restriction base="xs:string">
      <xs:enumeration value="Divest"/>
      <xs:enumeration value="Acquire"/>
      <xs:enumeration value="DivestAcquire"/>
      <xs:enumeration value="NoTransfer"/>
    </xs:restriction>
  </xs:annotation>
</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:simpleType name="updateRateType">
  <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:sequence>
    <xs:any namespace="#other" minOccurs="0"/>
  </xs:complexType>
</xs:complexType>

<federate xsi:documentation>
  <xs:documentation>RTI services implemented/used in the federation or by a</xs:documentation>
  <xs:documentation>---</xs:documentation>
</federate>
SISO-STD-003-2006, BOM Template Specification

<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="requestAttributeOwnershipAssumption" minOccurs="0"/>
<xs:element name="confirmDivestiture" minOccurs="0"/>
<xs:element name="requestDivestitureConfirmation" minOccurs="0"/>
<xs:element name="attributeOwnershipAcquisitionNotification" minOccurs="0"/>
<xs:element name="attributeOwnershipAssumptionAvailable" minOccurs="0"/>
<xs:element name="attributeOwnershipUnavailable" minOccurs="0"/>
<xs:element name="requestAttributeOwnershipRelease" minOccurs="0"/>
<xs:element name="attributeOwnershipReleaseDenied" minOccurs="0"/>
<xs:element name="cancelNegotiatedAttributeOwnershipDivestiture" minOccurs="0"/>
<xs:element name="cancelAttributeOwnershipAcquisitionCancellation" minOccurs="0"/>
<xs:element name="queryAttributeOwnership" minOccurs="0"/>
<xs:element name="informAttributeOwnership" minOccurs="0"/>
<xs:element name="isAttributeOwnedByFederate" minOccurs="0"/>
<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="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceNotification" minOccurs="0"/>
<xs:element name="timeAdvanceAvailable" minOccurs="0"/>
<xs:element name="nextMessageRequestAvailable" minOccurs="0"/>
<xs:element name="flushQueueRequest" minOccurs="0"/>
<xs:element name="timeAdvanceRequestAvailable" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
<xs:element name="timeAdvanceGrant" minOccurs="0"/>
<xs:element name="timeAdvanceRequest" minOccurs="0"/>
SISO-STD-003-2006, BOM Template Specification

<xs:element name="getDimensionUpperBound" minOccurs="0"/>
<xs:element name="getAvailableDimensionsForClassAttribute" minOccurs="0"/>
<xs:element name="getKnownObjectHandle" minOccurs="0"/>
<xs:element name="getAvailableDimensionsForInteractionClass" 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="enableInteractionRelevanceAdvisorySwitch" minOccurs="0"/>
<xs:element name="disableInteractionRelevanceAdvisorySwitch" minOccurs="0"/>
<xs:element name="getDimensionHandleSet" 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="getFederateName" minOccurs="0"/>
<xs:element name="getUpdateRateValueForAttribute" minOccurs="0"/>
</xs:all>
<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:complexType>
<xs:all>
<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:sequence>
</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="[\d\]0]+([\d]*\[\d]+)\[([\d]*)\]([\d]*\[\d]+)\]*/
</xs:restriction>
</xs:simpleType>
<xs:complexType name="cardinalityType">
<xs:complexContent>
<xs:extension base="cardinalityPattern">
<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:extension>
</xs:complexContent>
</xs:simpleType>
<xs:simpleType name="arrayDatatypeEncodingUnion">
<xs:restriction base="xs:string">
<xs:pattern value="HLAfixedArray"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="arrayDatatypeEncodingEnum">
<xs:restriction base="xs:string">
<xs:pattern value="HLAvariableArray"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="arrayDatatypeEncodingUnion modelID:nonEmptyString"/>
<xs:simpleType name="arrayDatatypeEncodingUnion">
<xs:restriction base="arrayDatatypeEncodingUnion">
<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="fixedRecordEncodingEnumeration">
<xs:restriction base="xs:string">
<xs:enumeration value="HLAfixedRecord"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="fixedRecordEncodingUnion">
<xs:restriction base="fixedRecordEncodingEnumeration modelID:nonEmptyString"/>
</xs:simpleType>

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
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
<?xml version="1.0"?>
<!-- edited with XMLSpy v2005 U (http://www.xmlspy.com) by Tram Chase (SimVentions)-->
<!-- edited with XMLSPY v6 rel. 4 U (http://www.xmlspy.com) by Bjorn Lofstrøm (Pitch Technologies)-->
<!-- Example BOM: The only namespaces needed apart from the standards-->
  <modelID:identification
    <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:identification>
  <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 44 123-456-1000</modelID:pocTelephone>
    <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:pocTelephone>+1 44 123-456-7890</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://bons.info/restaurantconceptualmodel.doc</modelID:identification>
  </modelID:reference>
  <modelID:reference>
    <modelID:other>See the pattern data model</modelID:other>
    <modelID:identification>
      <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>WaiterEntity</receiver>
              <semantics>Waiter observes that the table is full of dirty dishes</semantics>
            </variation>
          </action>
        </conceptualModelDefinition>
      </conceptualModelDefinition>
    </modelID:identification>
  </modelID:reference>
</BOM>
```

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
SISO-STD-003-2006, BOM Template Specification

<variation>
<variation>
  <name>CustomerRequest</name>
  <event>CheckRequest</event>
  <sender>CustomerEntity</sender>
  <receiver>WaiterEntity</receiver>
  <semantics>Waiter is called by customer</semantics>
</variation>
</variation>

<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>

<stateMachine>
  <name>EmployeeSM</name>
  <conceptualEntity>Waiter</conceptualEntity>
  <conceptualEntity>Customer</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>
    </exitCondition>
  </state>
</stateMachine>

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
<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>
</eventTypes>
<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>Notification of Payment Due</semantics>
</eventType>
<eventType>
  <name>Tip</name>
  <sourceCharacteristic>
    <name>CustomerEntity.ID</name>
  </sourceCharacteristic>
  <targetCharacteristic>
    <name>WaiterEntity.ID</name>
  </targetCharacteristic>
  <contentCharacteristic>
    <name>BillEntity.Amount</name>
  </contentCharacteristic>
  <semantics>Tip left for waiter by customer</semantics>
</eventType>
</conceptualModelDefinition>

<entityTypeMappings>
  <eventTypeMapping>
    <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>
  </eventTypeMapping>
  <eventTypeMapping>
    <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>
  </eventTypeMapping>
  <eventTypeMapping>
    <name>WaiterEntity</name>
    <hlaClass>HLAobjectRoot.PhysicalEntity.Human.Waiter</hlaClass>
    <characteristic>
    </characteristic>
  </eventTypeMapping>
</entityTypeMappings>
<eventTypeMappings>
  <eventTypeMapping>
    <name>PaymentDueNotification</name>
    <hlaClass>HLAInteractionRoot.RealWorldMessage.PaymentDue</hlaClass>
    <sourceCharacteristic>
      <name>Waiter_Entity.ID</name>
      <hlaProperty>PaymentDue.WaiterID</hlaProperty>
    </sourceCharacteristic>
    <targetCharacteristic>
      <name>Customer_Entity.ID</name>
      <hlaProperty>PaymentDue.CustomerID</hlaProperty>
    </targetCharacteristic>
  </eventTypeMapping>
</eventTypeMappings>

<eventTypeMappings>
  <eventTypeMapping>
    <name>NonPayingCustomer</name>
    <hlaClass>HLAInteractionRoot.PhysicalEvent.CustomerLeaves</hlaClass>
    <sourceCharacteristic>
      <name>Table_Entity.ID</name>
      <hlaProperty>HLAobjectRoot.PhysicalEntity.Table.ID</hlaProperty>
    </sourceCharacteristic>
    <triggerCondition>
      <name>Table_Entity.Dishes_Dirty.equals(TRUE)</name>
      <hlaProperty>Table.hasDirtyDishes == TRUE</hlaProperty>
    </triggerCondition>
  </eventTypeMapping>
</eventTypeMappings>

<eventTypeMappings>
  <eventTypeMapping>
    <name>CheckRequest</name>
    <hlaClass>HLAInteractionRoot.RealWorldMessage.RequestBill</hlaClass>
    <sourceCharacteristic>
      <name>Customer_Entity.ID</name>
      <hlaProperty>RequestBill.CustomerID</hlaProperty>
    </sourceCharacteristic>
    <targetCharacteristic>
      <name>Waiter_Entity.ID</name>
      <hlaProperty>RequestBill.WaiterID</hlaProperty>
    </targetCharacteristic>
  </eventTypeMapping>
</eventTypeMappings>

<eventTypeMappings>
  <eventTypeMapping>
    <name>DirtyDishesOnTable</name>
    <hlaClass>HLAobjectRoot.PhysicalEntity.Table</hlaClass>
    <sourceCharacteristic>
      <name>Table_Entity.ID</name>
      <hlaProperty>Table.ID</hlaProperty>
    </sourceCharacteristic>
    <triggerCondition>
      <name>Table_Entity.Dishes_Dirty.equals(TRUE)</name>
      <hlaProperty>Table.hasDirtyDishes == TRUE</hlaProperty>
    </triggerCondition>
  </eventTypeMapping>
</eventTypeMappings>

<eventTypeMappings>
  <eventTypeMapping>
    <name>BillEntity</name>
    <hlaClass>HLAobjectRoot.PhysicalEntity.Payment</hlaClass>
    <sourceCharacteristic>
      <name>Amount</name>
      <hlaProperty>Payment.Amount</hlaProperty>
    </sourceCharacteristic>
    <characteristic>
      <name>Tip</name>
      <hlaProperty>Payment.Tip</hlaProperty>
    </characteristic>
    <characteristic>
      <name>Waiter_ID</name>
      <hlaProperty>Waiter.ID</hlaProperty>
    </characteristic>
    <characteristic>
      <name>CustomerAssignedTableID</name>
      <hlaProperty>Customer.AssignedTableID</hlaProperty>
    </characteristic>
    <characteristic>
      <name>Waiter_Amount</name>
      <hlaProperty>Waiter.Amount</hlaProperty>
    </characteristic>
    <characteristic>
      <name>Name</name>
      <hlaProperty>Waiter_Name</hlaProperty>
    </characteristic>
  </eventTypeMapping>
</eventTypeMappings>

<eventTypeMappings>
  <eventTypeMapping>
    <name>Assigned_Tables</name>
    <hlaProperty>Waiter.AssignedTables</hlaProperty>
  </eventTypeMapping>
</eventTypeMappings>

<eventTypeMappings>
  <eventTypeMapping>
    <name>Table</name>
    <hlaClass>HLAobjectRoot.PhysicalEntity.Table</hlaClass>
    <sourceCharacteristic>
      <name>ID</name>
      <hlaProperty>Table.ID</hlaProperty>
    </sourceCharacteristic>
    <characteristic>
      <name>Dishes_Dirty_equals_TRUE</name>
      <hlaProperty>Table.hasDirtyDishes == TRUE</hlaProperty>
    </characteristic>
    <characteristic>
      <name>Assigned_Tables</name>
      <hlaProperty>Table.Assigned_Tables</hlaProperty>
    </characteristic>
    <characteristic>
      <name>Waiter_ID</name>
      <hlaProperty>Table.Waiter_ID</hlaProperty>
    </characteristic>
    <characteristic>
      <name>Name</name>
      <hlaProperty>Table.Name</hlaProperty>
    </characteristic>
  </eventTypeMapping>
</eventTypeMappings>

<eventTypeMappings>
  <eventTypeMapping>
    <name>CheckRequest</name>
    <hlaClass>HLAInteractionRoot.RealWorldMessage.RequestBill</hlaClass>
    <sourceCharacteristic>
      <name>Customer_Entity.ID</name>
      <hlaProperty>RequestBill.CustomerID</hlaProperty>
    </sourceCharacteristic>
    <targetCharacteristic>
      <name>Waiter_Entity.ID</name>
      <hlaProperty>RequestBill.WaiterID</hlaProperty>
    </targetCharacteristic>
  </eventTypeMapping>
</eventTypeMappings>

<eventTypeMappings>
  <eventTypeMapping>
    <name>PaymentDueNotification</name>
    <hlaClass>HLAInteractionRoot.RealWorldMessage.PaymentDue</hlaClass>
    <sourceCharacteristic>
      <name>Waiter_Entity.ID</name>
      <hlaProperty>PaymentDue.WaiterID</hlaProperty>
    </sourceCharacteristic>
    <targetCharacteristic>
      <name>Customer_Entity.ID</name>
      <hlaProperty>PaymentDue.CustomerID</hlaProperty>
    </targetCharacteristic>
  </eventTypeMapping>
</eventTypeMappings>

<eventTypeMappings>
  <eventTypeMapping>
    <name>DirtyDishesOnTable</name>
    <hlaClass>HLAobjectRoot.PhysicalEntity.Table</hlaClass>
    <sourceCharacteristic>
      <name>ID</name>
      <hlaProperty>Table.ID</hlaProperty>
    </sourceCharacteristic>
    <sourceCharacteristic>
      <name>Dishes_Dirty_equals_TRUE</name>
      <hlaProperty>Table.hasDirtyDishes == TRUE</hlaProperty>
    </sourceCharacteristic>
  </eventTypeMapping>
</eventTypeMappings>

<eventTypeMappings>
  <eventTypeMapping>
    <name>BillEntity</name>
    <hlaClass>HLAobjectRoot.PhysicalEntity.Payment</hlaClass>
    <sourceCharacteristic>
      <name>Amount</name>
      <hlaProperty>Payment.Amount</hlaProperty>
    </sourceCharacteristic>
    <characteristic>
      <name>Tip</name>
      <hlaProperty>Payment.Tip</hlaProperty>
    </characteristic>
    <characteristic>
      <name>Waiter_ID</name>
      <hlaProperty>Waiter.ID</hlaProperty>
    </characteristic>
  </eventTypeMapping>
</eventTypeMappings>

<eventTypeMappings>
  <eventTypeMapping>
    <name>Assigned_Tables</name>
    <hlaProperty>Waiter.AssignedTables</hlaProperty>
  </eventTypeMapping>
</eventTypeMappings>

<eventTypeMappings>
  <eventTypeMapping>
    <name>Table</name>
    <hlaClass>HLAobjectRoot.PhysicalEntity.Table</hlaClass>
    <sourceCharacteristic>
      <name>ID</name>
      <hlaProperty>Table.ID</hlaProperty>
    </sourceCharacteristic>
    <characteristic>
      <name>Dishes_Dirty_equals_TRUE</name>
      <hlaProperty>Table.hasDirtyDishes == TRUE</hlaProperty>
    </characteristic>
    <characteristic>
      <name>Assigned_Tables</name>
      <hlaProperty>Table.Assigned_Tables</hlaProperty>
    </characteristic>
    <characteristic>
      <name>Waiter_ID</name>
      <hlaProperty>Table.Waiter_ID</hlaProperty>
    </characteristic>
    <characteristic>
      <name>Name</name>
      <hlaProperty>Table.Name</hlaProperty>
    </characteristic>
  </eventTypeMapping>
</eventTypeMappings>
PublishSubscribe

Represents a payment using cash.

Amount

Represents a payment using cash.*

Tip

Represents a payment using cash.*

Source

Represents a payment using cash.*
This is an approved SISO Standard.

This pattern of interplay might be able to be used as a template to support payment patterns for other service oriented businesses. This pattern action may involve other employees including the host (i.e. Greeter) who is the idle state for the Greeter or Waiter.

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Country" /></td>
<td>Mandatory single element</td>
<td>A rectangle with a solid border identifies a required XML schema element. In this example, a mandatory element of Country is required to be identified within the XML instance data.</td>
</tr>
<tr>
<td><img src="image" alt="Name" /></td>
<td>Mandatory single element containing parsed character data (i.e., a child text node)</td>
<td>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). In 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.</td>
</tr>
<tr>
<td><img src="image" alt="Location" /></td>
<td>Single optional element</td>
<td>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.</td>
</tr>
<tr>
<td><img src="image" alt="Alias" /></td>
<td>Mandatory multiple element</td>
<td>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.</td>
</tr>
<tr>
<td><img src="image" alt="Division" /></td>
<td>Mandatory multiple element containing child elements</td>
<td>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.</td>
</tr>
<tr>
<td><img src="image" alt="xs:field" /></td>
<td>Element referencing global element</td>
<td>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.</td>
</tr>
<tr>
<td><img src="image" alt="keybase" /></td>
<td>Complex Type</td>
<td>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.</td>
</tr>
<tr>
<td><img src="image" alt="Subsidiaries" /></td>
<td>Model Group</td>
<td>An irregular octagon with a plus sign indicates a model group, which can be used to define and reuse element declarations. In this example, Subsidiaries is a global model group that can be used to define elements within the XML schema. reuse element declarations.</td>
</tr>
<tr>
<td><img src="image" alt="any ## other" /></td>
<td>Wildcards</td>
<td>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:</td>
</tr>
</tbody>
</table>

---

Copyright © 2006 SISO. All rights reserved.
This is an approved SISO Standard.
### 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](image)

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>
<thead>
<tr>
<th>Compositor</th>
<th>Diagram</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td><img src="image" alt="Sequence Diagram" /></td>
<td>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.</td>
</tr>
<tr>
<td>Choice</td>
<td><img src="image" alt="Choice Diagram" /></td>
<td>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 unlimited number of Division instances.</td>
</tr>
<tr>
<td>All</td>
<td><img src="image" alt="All Diagram" /></td>
<td>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.</td>
</tr>
</tbody>
</table>
This page left intentionally blank