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 January 15, 2003. This document would not have been possible without the hard work and dedicated efforts of the following individuals:

Drafting Group Team
Lead Editor: Paul Gustavson pgustavson@simventions.com
   Jake Borah jborah@aegisg.com
   Tram Chase tchase@simventions.com
   Stephen M. Goss sgoss@simventions.com
   Björn Löfstrand bjorn.lofstrand@pitch.se
   Steven W. Reichenthal steven.w.reichenthal@boeing.com
   Roy Scrudder scrudder@arlut.utexas.edu
   Chris Turrell cturrell@msiac.dmso.mil

Additional Author Contributions
   Bob Lutz robert.lutz@jhuapl.edu

Product Development Group Officers
PDG Chair: Lawrence M. Root lroot@simventions.com
PDG Vice-Chair: Bob Lutz robert.lutz@jhuapl.edu
PDG Secretary: Jane T. Bachman Jane.Bachman@navy.mil
SAC TAD: Chris Rouget cjrouget@preforce.demon.co.uk

Assigned Reviewers
   Mark Biwer mbiwer@cntr.dtra.mil
   George Hughes ghughes@simventions.com
   Reed Little little@sei.cmu.edu
   Staffan Löf staffan.lof@pitch.se
   Katherine Morse katherine.l.morse@saic.com

Technical Editing
   Troy Sizemore tsizemore@simventions.com

Copyright © 2004-2005 SISO. All rights reserved.
This is an unapproved SISO Draft Specification, subject to change.
# 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** ............................................................................................................................ 22  
      6.1.1 **Purpose/Background** ............................................................................................................................ 22  
      6.1.2 **Table Format** ......................................................................................................................................... 22  
      6.1.3 **Inclusion Criteria** .................................................................................................................................. 24  
      6.1.4 **Example** ................................................................................................................................................. 25  
   6.2 **CONCEPTUAL MODEL** ................................................................................................................................. 26  
      6.2.1 **Pattern Description** .................................................................................................................................. 26  
      6.2.1.1 **Purpose/Background** .......................................................................................................................... 26  
      6.2.1.2 **Table Format** ....................................................................................................................................... 26  
      6.2.1.3 **Inclusion Criteria** .................................................................................................................................. 28  
      6.2.1.4 **Example** ................................................................................................................................................. 28  
      6.2.2 **State Machine Description** .................................................................................................................... 30  
      6.2.2.1 **Purpose/Background** .......................................................................................................................... 30  
      6.2.2.2 **Table Format** ....................................................................................................................................... 30  
      6.2.2.3 **Inclusion Criteria** .................................................................................................................................. 31  
      6.2.2.4 **Example** ................................................................................................................................................. 31  
      6.2.3 **Entity Type** ............................................................................................................................................ 33  
      6.2.3.1 **Purpose/Background** .......................................................................................................................... 33  
      6.2.3.2 **Table Format** ....................................................................................................................................... 33  
      6.2.3.3 **Inclusion Criteria** .................................................................................................................................. 34  
      6.2.3.4 **Example** ................................................................................................................................................. 34  
      6.2.4 **Event Type** ........................................................................................................................................... 35  
      6.2.4.1 **Purpose/Background** .......................................................................................................................... 35  
      6.2.4.1.1 **BOM Triggers** ................................................................................................................................... 35  
      6.2.4.1.2 **BOM Messages** .................................................................................................................................. 35  
      6.2.4.2 **Table Format** ....................................................................................................................................... 36  
      6.2.4.3 **Inclusion Criteria** .................................................................................................................................. 37  
      6.2.4.4 **Example** ................................................................................................................................................. 37  
      6.2.4.4.1 **BOM Trigger Example** ......................................................................................................................... 38  
      6.2.4.4.2 **BOM Message Example** ....................................................................................................................... 38  
   6.3 **MODEL MAPPING** ........................................................................................................................................ 39  
      6.3.1 **Entity Type Mapping** ............................................................................................................................ 39  
      6.3.1.1 **Purpose/Background** .......................................................................................................................... 39  
      6.3.1.2 **Table Format** ....................................................................................................................................... 39  
      6.3.1.3 **Inclusion Criteria** .................................................................................................................................. 41  
      6.3.1.4 **Example** ................................................................................................................................................. 41  
      6.3.2 **Event Type Mapping** ............................................................................................................................... 41  
      6.3.2.1 **Purpose/Background** .......................................................................................................................... 41  
      6.3.2.2 **Table Format** ....................................................................................................................................... 42  
      6.3.2.3 **Inclusion Criteria** .................................................................................................................................. 44  
      6.3.2.4 **Example** ................................................................................................................................................. 44  
   6.4 **OBJECT MODEL DEFINITION** .................................................................................................................. 45  
   6.5 **SUPPORTING TABLES** ............................................................................................................................... 46  
   6.5.1 **BOM Notes** ............................................................................................................................................... 46

Copyright © 2004-2005 SISO. All rights reserved. 
This is an unapproved SISO Draft Specification, subject to change.
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 and agile, rapid, and efficient development and maintenance of models as well as their integration into operational systems or embedding real-world systems into virtual environments.

Base Object Models (BOMs) are intended to provide a component framework for facilitating interoperability, reuse, and composability. The BOM concept is based on the assumption that piece-parts of 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 simulation interplay are sequences of events between simulation elements. The representation of the pattern is captured in the BOM.

There are two BOM related documents developed by the SISO BOM Product Development Group (PDG). These documents are the “BOM Template Specification” and the “BOM Guidance Specification.” This document is the “BOM Template Specification.”

1.1 Purpose

The 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 is a required component for enabling tools that can exchange and reason about BOMs.

1.2 Scope

The BOM Template Specification defines semantics and 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 simulation will need to do.

The required simulated capabilities which are reflected in the conceptual model can also be defined in the context of an interface description that represents the information necessary for execution and exchange. This interface information is often described in terms of structural and/or object-based classes defining 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 an interface description 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 elements of the Conceptual Model and the interface description 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 BOM Guidance Specification provides discussion on BOM development and the application and use of BOMs for the assembly of simulations and simulation spaces as illustrated in Figure 1-1.

The green rectangular region in Figure 1-1 represents a playing field, and is analogous to a simulation environment in which BOMs may be composed and used. The items marked A, B, C and X within this playing field 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 the end-state of a simulation conceptual model, and mapping the interface elements of a simulation component, which are described in terms of HLA OMT constructs. The objective is to encourage reuse, support composability, and help enable rapid development of simulations and simulation spaces. 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 systems and simulations. 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 following publications. If any of the specifications identified in the following two tables are superseded by an approved revision, then the revision shall apply.

Table 2-1 - SISO 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>N/A</td>
<td>BOM PDG Product Nomination – March 2003</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>XML 1.0 (Second Edition)</td>
<td>Extensible Markup Language (XML) 1.0 (Second Edition) W3C Recommendation, 6 October 2000</td>
</tr>
</tbody>
</table>
## 3 Definitions

The following terms are 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>
<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 composed of a group of interrelated elements, which can be used as a building block in the development and extension of a federation, individual federate, FOM or SOM. BOM elements include object classes, interaction classes, patterns of interplay, state machines, and events.</td>
</tr>
<tr>
<td>BOM Assembly</td>
<td>A composition of BOMs, which can result in a FOM, 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>A unit with a known set of inputs and expected output behavior, but the implementation details may be hidden.</td>
</tr>
<tr>
<td>Composability</td>
<td>&quot;The capability to select and assemble components in various combinations into complete, validated simulation environments to satisfy specific user requirements.&quot; These environments may support a variety of application domains, levels of resolution, and time scales.</td>
</tr>
<tr>
<td>Conceptual Entity</td>
<td>An abstract representation of a real world entity, phenomenon, process, or system.</td>
</tr>
<tr>
<td>Conceptual Event</td>
<td>A representation of a transient action that occurs among conceptual entities which may affect the state of one or more of the conceptual entities.</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>Refers to a simulation, an interface to a live system, or a supporting utility such as a Logger, Plan View Display, or Stealth Viewer. 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>
<tr>
<td>Federation</td>
<td>A collection of one or more federates capable of interoperating within a distributed synthetic environment. In HLA, &quot;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.&quot; Such a FOM can be a result of a BOM Assembly.</td>
</tr>
</tbody>
</table>

---

| **Glyph** | An image used for the visual representation of a BOM such as within a tool palette or web page. |
| **Message** | An event, which is directed to a specific receiver(s). |
| **Metadata** | “Structured, encoded data that describe characteristics of information-bearing entities to aid in the identification, discovery, assessment, and management of the described entities”5. |
| **Pattern** | A named set of recurring behavior used to accomplish a common objective, capability or purpose. |
| **Pattern of Interplay** | Specific type of pattern characterized by a sequence of activities involving one or more conceptual entities. |
| **Purpose** | Identifies the end or aim to which a BOM is intended. |
| **Simulation Space** | Any part of a simulation environment, which may be represented by one or more conceptual entities modeled and reflected by a federate, federation, or as an aggregate entity within a federation execution. |
| **State Machine** | A description of the various states or conditions of a conceptual entity, and how the actions associated with one or more patterns of interplay may affect these conditions over its life. |
| **Trigger** | An event, which is not directed to a specific receiver(s). |

---

5 The Final Report of the Association for Library Collections and Technical Services’ Task Force on Metadata (2000), Copyright © 2004-2005 SISO. All rights reserved. This is an unapproved SISO Draft Specification, subject to change.
# 4 Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</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>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>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>RPR FOM</td>
<td>Real-time Platform Reference FOM</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>UML</td>
<td>Unified Modeling 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 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 names
- Pattern names
- Pattern Action names
- State Machine names
- State names
- Entity Type names
- Event Type names
- Object Class names
- Interaction Class names
- Attribute names
- Parameter names
- Datatype names
- Enumerated datatype enumerators
- Enumerated datatype values
- Fixed Record Field names
- Variant Record Alternative names
- Basic Data Representation names
- Note identifying labels

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 object class of interest, and the turret as the object class attribute that is of interest. Both the M1 and turret are easily referenced by the hierarchy tree defined in the text.

Although individual names need not be unique, all entity types, event types, pattern actions, object classes, and event classes within a BOM shall be uniquely identifiable when concatenated (via Dot Notation) with the names of higher level superclasses. If an action, characteristic, attribute or 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, interface information defined using HLA OMT constructs, and the mapping between conceptual model elements and object model definition elements, which are used to represent the interface information. Figure 6-1 provides an illustration of the BOM template, which is made of four template components: Model Identification, Conceptual Model, Model Mapping and Object Model Definition.

Figure 6-1 - BOM Composition

The BOM template components extend the semantics and syntax of the HLA OMT by specifying the pattern of interplay, state machine, types of entities and events at the conceptual model level of description and
provide a mechanism for mapping the conceptual model elements to object model definition elements, which are described using HLA OMT constructs.

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 here is designed for presentation on a printed page, whereas the BOM DIF (as defined in Section 6) 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 template components taken directly from the IEEE Std 1516.2-2000 OMT Specification, which hereinafter will be referred to as the OMT Specification, and additional template components defined in this document. The set of template components used for representing BOMs are identified in Table 6-1. The template components leveraged from the OMT Specification are not redefined here. The application of the tables in the list below differs from that in the OMT Specification in that they are used for specifying aspects of the conceptual model and documenting an interface of a simulation component, which can be used and reused in the design, development and extension of interoperable simulations. The object class, interaction class, attributes, parameters, datatypes, notes, and object model lexicon template components follow the same structure in a BOM as they do in an HLA Federation Object Model (FOM) or Simulation Object Model (SOM) as defined in the OMT Specification. The use of these object model definition elements provide an external public view of the capabilities that can be supported or represented by federation participants thereby defining an interface, which can be mapped to views of a conceptual model. The views of a conceptual model are supported using template components defined exclusively within this specification. This includes pattern descriptions, state machines, entity types and event types. The model mappings are also supported using template components defined exclusively within this specification. This includes an Entity Mapping Table and Event Mapping Table.

**Table 6-1 – BOM Tables**

<table>
<thead>
<tr>
<th>Category</th>
<th>Presentation Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Identification</td>
<td>Model Identification Table</td>
<td>To associate important identifying information with the BOM. This is an extension of the Object Model Identification Table found in the OMT Specification.</td>
</tr>
<tr>
<td>Metadata</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conceptual Model</td>
<td>Patterns Description Table</td>
<td>To identify the pattern actions including variations and exceptions required for supporting the activities associated with a pattern of interplay. This is an extension of the OMT Specification, which provides a pattern description associated with the event types that can be identified in a BOM.</td>
</tr>
<tr>
<td>State Machine Table</td>
<td></td>
<td>To identify the conceptual entity states required for supporting the activities associated with a pattern of interplay. This is an extension of the OMT Specification, which provides a state machine model associated with the entity types that can be identified in a BOM.</td>
</tr>
<tr>
<td>Entity Type Table</td>
<td></td>
<td>To identify the types of entities required for supporting the aspects of the conceptual model.</td>
</tr>
<tr>
<td>Event Type Table</td>
<td></td>
<td>To identify the types of events, whether directed (messages) or undirected events (triggers), required for supporting the actions associated with a pattern of interplay.</td>
</tr>
<tr>
<td>Model Mapping</td>
<td>Entity Type Mapping Table</td>
<td>To map what HLA constructs can be used to represent the entity types identified in the conceptual model of a BOM.</td>
</tr>
</tbody>
</table>
### Event Type Mapping Table
To identify and associate what HLA constructs can be used to represent the various event types identified in the conceptual model of a BOM.

<table>
<thead>
<tr>
<th>Event Type Mapping Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Model Definition</td>
<td>Object Class Structure Table</td>
</tr>
<tr>
<td></td>
<td>Interaction Class Structure Table</td>
</tr>
<tr>
<td></td>
<td>Attribute Table</td>
</tr>
<tr>
<td></td>
<td>Parameter Table</td>
</tr>
<tr>
<td></td>
<td>Datatype Tables</td>
</tr>
<tr>
<td>Supporting Tables</td>
<td>Notes Table</td>
</tr>
<tr>
<td></td>
<td>Lexicon Tables</td>
</tr>
</tbody>
</table>

Unless noted otherwise in the "Guide for 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. The specific rules for the applicability of each BOM template component are provided in the table descriptions.

The last BOM template component, the Lexicon Tables, is essential to ensure that the semantics of the terms used in a BOM are understood and documented.

Any BOM which fully conforms to all of the rules and constraints stated in this specification shall be considered a compliant base object model.

The basics of each BOM template component are presented in the following separate sub-clauses. The format for each template component is provided and described, and criteria are suggested to help guide decisions on completing specific categories within each of these template components. Example use of each template component is also provided; these examples are for illustrative purposes only and are not meant to imply any additional requirements.

The BOM DIF tables presented in Tables 6-3, 6-6, 6-9, 6-12, 6-15, 6-18 and 6-21 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>). Optional information is enclosed in square brackets (e.g., [<limitation>]). Fields that support optional information, but have no value for a specific table instance, should be filled with "na."

---

6.1 Model Identification

6.1.1 Purpose/Background

A design goal for all BOMs is to facilitate reuse. BOMs provide information that enable inferences to be drawn regarding the reuse potential of individual patterns for new applications. 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 Table is to document certain key metadata information about the BOM.

6.1.2 Table Format

Table 6-2 provides a description of the categories of information that comprise a Model Identification. Many of the information categories used for the Model Identification are leveraged from the 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 6-2. Italics are used in the Values column of Table 6-2 to denote the type of data to 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>IEEE Std 1516.2-2000</th>
<th>Occurs</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Specifies the name assigned to the object model.</td>
<td>yes</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 &quot;BOM.&quot;</td>
<td>yes</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>yes</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 object 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 &quot;yyyy-mm-dd&quot; (e.g., 1999-04-15).</td>
<td>yes</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>yes</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>new</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>yes</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 pattern pertains.</td>
<td>yes</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>new</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>new</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>new</td>
<td>0..many</td>
<td>text</td>
</tr>
<tr>
<td>Keyword</td>
<td>This pair of fields shall be repeated as many times as necessary.</td>
<td>new</td>
<td>0..many</td>
<td></td>
</tr>
<tr>
<td>Taxonomy</td>
<td>This field specifies the source of the keyword vocabulary.</td>
<td>new</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>new</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>Category</td>
<td>Description</td>
<td>IEEE Std 1516.2-2000</td>
<td>Occurs</td>
<td>Values</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------</td>
<td>--------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>POC</td>
<td>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 must be supplied. Multiple sets may be supplied.</td>
<td>allow multiple</td>
<td>1..many</td>
<td></td>
</tr>
<tr>
<td>POC Type</td>
<td>This field shall specify the role that the POC has with respect to the BOM.</td>
<td>new</td>
<td>1</td>
<td>Primary author, Contributor, Proponent, Sponsor, Release authority, Technical POC, other text</td>
</tr>
<tr>
<td>POC Name</td>
<td>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 must be specified.</td>
<td>identified as ‘POC’</td>
<td>0..1</td>
<td>Text</td>
</tr>
<tr>
<td>POC Organization</td>
<td>This field shall specify the organization if 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.</td>
<td>yes</td>
<td>0..1</td>
<td>Text</td>
</tr>
<tr>
<td>POC Telephone</td>
<td>This field shall specify the telephone number for the POC including the international telephone code for the POC’s country.</td>
<td>yes</td>
<td>0..many</td>
<td>Text</td>
</tr>
<tr>
<td>POC Email</td>
<td>This field shall specify the email address of the POC.</td>
<td>yes</td>
<td>1..many</td>
<td>text</td>
</tr>
<tr>
<td>Reference</td>
<td>This set of fields shall specify a pointer to additional sources of information.</td>
<td>allow multiple</td>
<td>0..many</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>This field shall specify the way in which the reference is related to the BOM.</td>
<td>new</td>
<td>1</td>
<td>Source material, Conceptual model, Related BOM, other text</td>
</tr>
<tr>
<td>Identification</td>
<td>This field shall specify how to locate the reference source. Examples include a Uniform Resource Identifier (URI), XML reference ID (ref ID), or ISBN.</td>
<td>new</td>
<td>1</td>
<td>Text</td>
</tr>
<tr>
<td>Other</td>
<td>This field shall specify other data deemed relevant by the author of the object model.</td>
<td>yes</td>
<td>0..1</td>
<td>Text</td>
</tr>
<tr>
<td>Glyph</td>
<td>This field shall specify a glyph to visually represent a BOM in a tool palette or web-based repository.</td>
<td>new</td>
<td>0..1</td>
<td></td>
</tr>
<tr>
<td>Image</td>
<td>This field holds the image representing the BOM.</td>
<td>new</td>
<td>1</td>
<td>base64</td>
</tr>
<tr>
<td>Type</td>
<td>This field holds the image type being represented.</td>
<td>new</td>
<td>1</td>
<td>BMP, GIF, JPG, PNG, TIFF, other text</td>
</tr>
<tr>
<td>Alt</td>
<td>This field shall be used to provide alternative text in case the image represented in the Image field cannot be displayed.</td>
<td>new</td>
<td>0..1</td>
<td>Text</td>
</tr>
<tr>
<td>Height</td>
<td>This field shall specify the pixel height of the glyph image represented in the Image field.</td>
<td>new</td>
<td>0..1</td>
<td>Short</td>
</tr>
<tr>
<td>Width</td>
<td>This field shall specify the pixel width of the glyph image represented in the Image field.</td>
<td>new</td>
<td>0..1</td>
<td>Short</td>
</tr>
</tbody>
</table>

The Model Identification Table is the template component of the BOM DIF used to identify the basic model metadata of a BOM and is provided in Table 6-3.

Table 6-3 Model Identification Table

<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>
</tbody>
</table>
The first column (Category) specifies the categories of data that shall be provided in this table.

The second column (Information) specifies the required information. Entries shall be required for all rows except those where the information is shown in brackets. Unless otherwise stated in the definitions, each row shall appear only once.

 Lexicon Definitions and Notes, which are not reflected in this view, can also be identified for these elements.

### 6.1.3 Inclusion Criteria

Every BOM shall have a Model Identification Table. The categories of information specified in Table 6-3 shall be included for all BOMs unless “0..1” or “0..many” is identified in the Occurs column of Table 6-2, in which it is then optional.
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>Purpose</td>
<td>Standardize the pattern of interplay between the 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 BOM Guidance and BOM 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>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 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://bons.info/restaurantconceptualmodel.doc">http://bons.info/restaurantconceptualmodel.doc</a></td>
</tr>
<tr>
<td>Other</td>
<td>This BOM pattern was featured on the Food Network special &quot;HLA and the Future of Restaurant Operations&quot;</td>
</tr>
<tr>
<td>Glyph</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>JPG</td>
</tr>
<tr>
<td>All</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>1</td>
</tr>
</tbody>
</table>

6.2 Conceptual Model

Since the BOM can be used to represent the end-state of 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 what simulation conceptual elements identified as entity types and event types are used. This section describes the various mechanisms for capturing the elements of a conceptual model that can be represented by a BOM.
6.2.1 Pattern Description

6.2.1.1 Purpose/Background

Pattern Description provides a mechanism for identifying sequences of actions (including variations and exceptions) necessary for fulfilling the pattern of interplay, which may be represented by a BOM. The activities for a pattern action can be supported 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 also supports the ability to identify exceptions and variations that can be associated with an action. An exception is an unexpected action that may occur, which typically causes the remaining sequences of the pattern to fail (where the next action is not performed). A variation, however, identifies a different way for the action to be accomplished without impeding the completion and success of the pattern. The actions of a pattern may be used by the State Machine Table (see Section 6.2.2) to identify the behavior condition required to facilitate the transition from one state to the next.

6.2.1.2 Table Format

As depicted in Table 6-5, the pattern data type is intended to allow one or more actions to be defined including exceptions and variations, the types of conceptual entities involved in sending and receiving each action to be defined, and the BOM event types or other BOMs used for fulfilling the activities of an action to be defined.

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Occurs</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Identifies the name of the pattern.</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>Action</td>
<td>Identifies an action to be carried out by a BOM &quot;event&quot; or potentially another BOM. Collection of actions forms the scenario needed to successfully accomplish pattern of interplay.</td>
<td>1..many</td>
<td></td>
</tr>
<tr>
<td>Sequence</td>
<td>Identifies the sequence order of the action</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>Name</td>
<td>Identifies the name of the action</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>Sender</td>
<td>Identifies the conceptual entity type(s) responsible for sending the 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 action</td>
<td>1..many</td>
<td>Text</td>
</tr>
<tr>
<td>Event</td>
<td>Used to identify event type that supports the activity associated with this action</td>
<td>0..1</td>
<td>eventIdentifier</td>
</tr>
<tr>
<td>BOM</td>
<td>Used to identify another BOM that supports the activity associated with this action</td>
<td>0..1</td>
<td>bomIdentifier</td>
</tr>
<tr>
<td>Exception</td>
<td>Identifies unexpected but potential behavior</td>
<td>0..many</td>
<td></td>
</tr>
<tr>
<td>Sequence</td>
<td>Identifies the sequence order of the exception</td>
<td>0..1</td>
<td>Text</td>
</tr>
<tr>
<td>Name</td>
<td>Identifies the name of the action exception</td>
<td>1</td>
<td>Text</td>
</tr>
<tr>
<td>Sender</td>
<td>Identifies the conceptual entity type(s) intended to receive the action (event or BOM) representative of the exception</td>
<td>1..many</td>
<td>Text</td>
</tr>
<tr>
<td>Receiver</td>
<td>Used to identify event type that supports the activity associated with this action exception</td>
<td>0..many</td>
<td>Text</td>
</tr>
<tr>
<td>Event</td>
<td>Used to identify another BOM that supports the activity associated with this action 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 that the action can be accomplished</td>
<td>0..many</td>
<td></td>
</tr>
<tr>
<td>Sequence</td>
<td>Identifies the sequence order of the variation</td>
<td>0..1</td>
<td>Text</td>
</tr>
<tr>
<td>Name</td>
<td>Identifies the name of the action variation</td>
<td>1</td>
<td>Text</td>
</tr>
</tbody>
</table>
The Pattern Description Table is the template component of the BOM DIF used to identify patterns of a Conceptual Model within a BOM and is provided in Table 6-6.

Table 6-6 Pattern Description Table

<table>
<thead>
<tr>
<th>Category</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern</td>
<td>&lt;pattern name&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<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>Action</td>
<td>&lt;sequence&gt;</td>
<td>&lt;name&gt;</td>
<td>&lt;sender&gt;</td>
<td>&lt;receiver&gt;</td>
<td>[&lt;event&gt;]</td>
<td>[&lt;BOM&gt;]</td>
</tr>
<tr>
<td>Exception</td>
<td>[&lt;sequence&gt;]</td>
<td>&lt;name&gt;</td>
<td>&lt;sender&gt;</td>
<td>[&lt;receiver&gt;]</td>
<td>[&lt;event&gt;]</td>
<td>[&lt;BOM&gt;]</td>
</tr>
<tr>
<td>Variation</td>
<td>[&lt;sequence&gt;]</td>
<td>&lt;name&gt;</td>
<td>&lt;sender&gt;</td>
<td>&lt;receiver&gt;</td>
<td>[&lt;event&gt;]</td>
<td>[&lt;BOM&gt;]</td>
</tr>
</tbody>
</table>

- The first column (Category) specifies the categories of data that shall be provided in this table.
- The second column (Information) specifies the required information. Entries shall be required for all rows and cells except those where the information is shown in brackets.

Lexicon Definitions and Notes, which are not reflected in this view, can also be identified for these elements.

The relationship of the pattern sub-elements within a BOM is illustrated using UML as shown in Figure 6-2.
This figure illustrates how an action associated with the pattern that a BOM is representing can have its activity linked to either a specific BOM event type, or an entirely different BOM. Section 6.2.4 describes the template components needed for identifying BOM events (as either triggers or messages).

A Dot Notation is used for the Event columns for Actions, Variations and Exceptions to ensure that the reference within the template component is unambiguous. Thus, Event names shall use Dot Notation as necessary, which may include the full parentage to uniquely identify the event type being used.

### 6.2.1.3 Inclusion Criteria

Every BOM describing elements of a conceptual model shall contain a Pattern Description Table or reference a Pattern Description Table captured within another BOM. The categories of information specified in Table 6-6 shall be included for all BOMs, which contain a Pattern Description Table, unless “0..1” or “0..many” is identified in the Occurs column of Table 6-5, in which it is then optional.

### 6.2.1.4 Example

Table 6-7 gives an example of a pattern of interplay representing the payment of a meal at a restaurant. Three actions are provided with variations and exceptions. The interface details of how each action is carried out are dependent upon an event and/or associated BOM linked with the action. Events are described in Section 6.2.4. While events are discussed in this specification, the actual behavior modeling required to perform the events that support the identified actions is outside the scope of a BOM.
### Table 6-7 - Pattern Description Table Example

<table>
<thead>
<tr>
<th>Category</th>
<th>Information</th>
<th>Name</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern</td>
<td>RestaurantPayment</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Seq</th>
<th>Name</th>
<th>Sender</th>
<th>Receiver</th>
<th>Event</th>
<th>BOM</th>
<th>Condition</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CustomerDone</td>
<td>Customer</td>
<td>Waiter</td>
<td>na</td>
<td>na</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variation</td>
<td>CustomerIdle</td>
<td>Table</td>
<td>Waiter</td>
<td>DirtyDishesOnTableTrigger</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variation</td>
<td>CustomerRequest</td>
<td>Customer</td>
<td>Waiter</td>
<td>CheckRequestMessage</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Action</td>
<td>CheckBroughtToTable</td>
<td>Waiter</td>
<td>Customer</td>
<td>PaymentDueNotificationMessage</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Action</td>
<td>CustomerPays</td>
<td>Customer</td>
<td>Waiter</td>
<td>na</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variation</td>
<td>BillPaided_Cash</td>
<td>Customer</td>
<td>Waiter</td>
<td>na</td>
<td>CashPaymentBOM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variation</td>
<td>BillPaided_Credit</td>
<td>Customer</td>
<td>Waiter</td>
<td>na</td>
<td>CreditCardPaymentBOM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Exception</td>
<td>BillNotPaided</td>
<td>Customer</td>
<td>na</td>
<td>NonPayingCustomerTrigger</td>
<td>NoPaymentBOM</td>
<td>customer leaves without paying bill</td>
</tr>
</tbody>
</table>
6.2.2 State Machine Description

6.2.2.1 Purpose/Background
State Machine Description provides a mechanism for identifying the behavior states required of a conceptual entity to support one or more patterns of interplay. This is achieved using a State Machine Table.

6.2.2.2 Table Format
As depicted in Table 6-8, the state machine description is intended to identify the states that can be represented behaviorally by one or more conceptual entities. A name and exit condition is identified for each state. Each exit condition is supported by an exit action that is associated with a pattern description (see Section 6.2.1), and identifies the next state upon satisfying the exit condition.

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Occurs</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<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 support the states defined</td>
<td>0..many</td>
<td>entity/identifier</td>
</tr>
<tr>
<td>State</td>
<td>Identifies a behavior state which a conceptual entity supports</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>Container that identifies the condition for a state transition. Includes the identification of the 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 action from a pattern in which the condition for exit has been satisfied. Note: an action from a pattern is supported by a BOM event or a BOM.</td>
<td>1</td>
<td>Text</td>
</tr>
<tr>
<td>Next State</td>
<td>Identifies what state succeeds this state based on the condition of the exit action.</td>
<td>1</td>
<td>Text</td>
</tr>
</tbody>
</table>

The State Machine Description Table is the template component of the BOM DIF used to identify state machines of a Conceptual Model within a BOM and is provided in Table 6-9.

- The first column (Category) specifies the categories of data that shall be provided in this table.
• The second column (Information) specifies the required information. Entries shall be required for all rows except those where the information is shown in brackets. Unless otherwise stated in the definitions, each row shall appear only once.

Lexicon Definitions and Notes, which are not reflected in this view, can also be identified for these elements.

The relationship of the state machine sub-elements within a BOM is illustrated using UML as shown in Figure 6-3.

![Figure 6-3 - BOM State Machine Relationship](image)

A Dot Notation is used for the Conceptual Entity and Exit Action columns to ensure that the reference within the template component is unambiguous. Thus, Entity Type 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 describing elements of a conceptual model shall contain a State Machine Table or reference a State Machine Table captured within another BOM. The categories of information specified in Table 6-9 shall be included for all BOMs, which contain a State Machine Table, unless “0..1” or “0..many” is identified in the Occurs column of Table 6-8, in which it is then optional.

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, Food Order. The interface details of what causes 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 pattern 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>Category</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Machine</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>EmployeeSM</td>
</tr>
<tr>
<td>Conceptual Entity</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>Greeter</td>
</tr>
<tr>
<td></td>
<td>Waiter</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>ExitCondition</th>
<th>ExitAction</th>
<th>Next State</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ready</td>
<td>RestaurantVisitationBOM.CustomerArrives</td>
<td>Greet</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RestaurantPaymentBOM.CustomerPays</td>
<td>ProcessBill</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greet</td>
<td>RestaurantVisitationBOM.GreetCustomer</td>
<td>Ready</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProcessOrder</td>
<td>FoodOrderBOM.SubmitOrderToKitchen</td>
<td>Ready</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PrepareBill</td>
<td>RestaurantPaymentBOM.CheckBroughtToTable</td>
<td>Ready</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProcessBill</td>
<td>RestaurantPaymentBOM.ReceiptChangeReturned</td>
<td>Ready</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ClearingTable</td>
<td>RestaurantVisitationBOM.CustomerLeft</td>
<td>Ready</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.2.3 Entity Type

6.2.3.1 Purpose/Background

Entity Types provide a mechanism for identifying the conceptual entity types required to support the patterns of interplay and fulfilling the various state machines. This is achieved using an Entity Type Table.

6.2.3.2 Table Format

As depicted in Table 6-11, the Entity Type is intended to identify entity types at the conceptual model. An Entity Type is used to support a sender or receiver identified within an action of a pattern description (see Section 6.2.1) and/or an entity associated with a State Machine (see Section 6.2.2). The entity type is identified by a name and associated characteristics.

Table 6-11 – Entity Type Information Categories

<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>text</td>
</tr>
</tbody>
</table>

The Entity Type Table is the template component of the BOM DIF used to identify entity types of a Conceptual Model within a BOM and is provided in Table 6-12. The Entity Type Table may identify one or more entity types.

Table 6-12 – Entity Type Table

<table>
<thead>
<tr>
<th>Category</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntityType</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>&lt;entity type name&gt;</td>
</tr>
<tr>
<td>Characteristic</td>
<td>&lt;characteristic&gt;</td>
</tr>
</tbody>
</table>

- The first column (Category) specifies the categories of data that shall be provided in this table.
- The second column (Information) specifies the required information. Entries shall be required for all rows except those where the information is shown in brackets. Unless otherwise stated in the definitions, each row shall appear only once.
- *Lexicon Definitions and Notes*, which are not reflected in this view, can also be identified for these elements.
- The characteristic(s) associated with an entity type within a BOM is illustrated using UML as shown in Figure 6-4.
6.2.3.3 Inclusion Criteria

Every BOM describing elements of a conceptual model shall contain an Entity Type Table or reference an Entity Type Table captured within another BOM. The categories of information specified in Table 6-11 shall be included for all BOMs which contain an Entity Type Table.

6.2.3.4 Example

Table 6-13 shows a simple example of how entity types and their characteristics are defined.

<table>
<thead>
<tr>
<th>Category</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Characteristics</td>
</tr>
<tr>
<td>Table</td>
<td>Has Dirty Dishes</td>
</tr>
<tr>
<td>Customer</td>
<td>Name</td>
</tr>
<tr>
<td>Waiter</td>
<td>Name</td>
</tr>
<tr>
<td>CashPayment</td>
<td>Amount</td>
</tr>
<tr>
<td></td>
<td>Waiter ID</td>
</tr>
<tr>
<td>Notes</td>
<td>Customer ID</td>
</tr>
</tbody>
</table>

Figure 6-4 - BOM Entity Type Relationship
6.2.4 Event Type

6.2.4.1 Purpose/Background

Event Type provides a mechanism for identifying the types of conceptual events used to represent and carry out the actions, variations, and exceptions within a pattern of interplay. The Event Type Table supports the ability to identify two types of BOM Events: BOM Triggers and BOM Messages.

6.2.4.1.1 BOM Triggers

When the condition (state) 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 entities within the virtual environment that have 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 supporting DoD and/or commercial projects.

A BOM Event that is a trigger is distinct because the conceptual entity identified as the sender of the action within the pattern of interplay does not direct the event to the conceptual entity identified as the receiver. This is because an event type that is a trigger, which is used to support a pattern action, identifies only the source entity identifier and the trigger condition, but not the target entity identifier. Thus, the key characteristic for a BOM Event that is a trigger is that the source conceptual entity is not made known of any target conceptual entities.

Typically a trigger occurs within an HLA execution when either an HLA Object Attribute has been updated or HLA Interaction has been sent, and the occurrence of that undirected event is of interest to a federate controlling an HLA object, which might react or respond to such an 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. A message is distinct because the conceptual entity identified as the sender of the action within the pattern of interplay directs the event to the conceptual entity identified as the receiver. This is because a BOM Event type that is a message identifies both source and target entity identifiers as characteristics.

Within an HLA execution, a message typically occurs between federates via an HLA Interaction Send or HLA Object Attribute Update. Specifically it is an event intended for a known type of conceptual entity; the conceptual entity is modeled in the HLA space by a federate in control of an HLA object, and, through the federate, the HLA object will react or respond to a specific HLA Interaction Send or HLA Object Attribute Update fulfilling the message event.

If multiple conceptual entities are knowledgeable of information either being passed or updated, then the identifier for each of the corresponding conceptual entities (both the source and affected target) and the item of interest being shared (the message contents being passed or attribute value being updated) should be classified as a type of BOM Event that is a message. However, if only one party of multiple conceptual entities cares about the information being passed or updated, then there would not be a conceptual entity identified as a target, only a source and a trigger condition, which would be classified together as a BOM Trigger as described in the previous section.
6.2.4.2 Table Format

As depicted in Table 6-14, the Event Type is intended to identify event types at the conceptual model. Event Type is used to support an event associated with a Pattern Description action (see Section 6.2.2). The event type is identified by a name, source, target and content characteristics and a trigger condition.

<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>SourceCharacteristic</td>
<td>Identifies the source characteristic for the event type</td>
<td>1</td>
<td>text</td>
</tr>
<tr>
<td>TargetCharacteristic</td>
<td>Identifies the target characteristic for the event type</td>
<td>0..many</td>
<td>text</td>
</tr>
<tr>
<td>ContentCharacteristic</td>
<td>Identifies the content characteristic for the event type</td>
<td>0..many</td>
<td>text</td>
</tr>
<tr>
<td>TriggerCondition</td>
<td>Identifies the trigger condition for the event type in the form of a Boolean expression.</td>
<td>0..1</td>
<td>text</td>
</tr>
</tbody>
</table>

The Event Type Table is the template component of the BOM DIF used to identify event types of a Conceptual Model within a BOM and is provided in Table 6-15. The Event Type Table may consist of one or more rows where each row represents an Event Type.

<table>
<thead>
<tr>
<th>Category</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Type</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>&lt;event type name&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;event type name&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;event type name&gt;</td>
</tr>
</tbody>
</table>

- The first column (Category) specifies the categories of data that shall be provided in this table. For this table, the category is strictly Event Type.

- The second column (Information) specifies the required information. Entries shall be required for all rows and cells except those where the information is shown in brackets. Unless otherwise stated in the definitions, each row shall appear only once.

  - The first Information column (Event Type Name) specifies a unique identifier for a BOM event type.

  - The second Information column (Source Characteristic) specifies the attribute-like qualities associated with the source of the event type.

  - The third Information column (Target Characteristic) specifies the attribute-like qualities associated with the target of the event type (optional).

  - The fourth Information column (Content Characteristic) specifies the attribute-like qualities associated with the content of the event type (optional).

  - The fifth Information column (Trigger Condition) specifies the trigger condition in the form of a Boolean expression associated with the event type (optional).
Lexicon Definitions and Notes, which are not reflected in this view, can also be identified for these elements. The relationship of the event type sub-elements within a BOM is illustrated using UML as shown in Figure 6-5.

![Diagram of Event Type Relationship](https://example.com/diagram.png)

**Figure 6-5 - BOM Event Type Relationship**

### 6.2.4.3 Inclusion Criteria

Every BOM describing elements of a conceptual model shall contain an Event Type Table or reference an Event Type Table captured within another BOM. The categories of information specified in Table 6-13 shall be included for all BOMs which contain an Event Type Table.

### 6.2.4.4 Example

Table 6-16 shows a simple example of how event types for both triggers and messages are defined. If the BOM developer marks a characteristic with a target role the event type could be said to be a "Message" and if there is no target but a trigger condition it is a "Trigger."

<table>
<thead>
<tr>
<th>Category</th>
<th>Name</th>
<th>Source Characteristic</th>
<th>Target Characteristic</th>
<th>Content Characteristic</th>
<th>Trigger Condition</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Type</td>
<td>DirtyDishesOnTable</td>
<td>Table_Identifier</td>
<td>na</td>
<td>na</td>
<td>Dishes_Dirty = TRUE</td>
<td></td>
</tr>
<tr>
<td>Event Type</td>
<td>NonPayingCustomer</td>
<td>Customer_identifier</td>
<td>na</td>
<td>na</td>
<td>Customer_NotPaid = TRUE</td>
<td>3</td>
</tr>
<tr>
<td>Event Type</td>
<td>CheckRequest</td>
<td>Customer_identifier</td>
<td>Waiter_identifier</td>
<td>na</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Event Type</td>
<td>PaymentDueNotification</td>
<td>Waiter_identifier</td>
<td>Customer_identifier</td>
<td>Amount</td>
<td></td>
<td>na</td>
</tr>
</tbody>
</table>

**Table 6-16 – Event Type Table Example**
6.2.4.4.1 BOM Trigger Example

A change in the condition or state of a conceptual entity can be used to trigger behavior in other conceptual entities. For instance, in the restaurant payment pattern example provided in Table 6-7, a change to the state of the conceptual entity identified as “Table,” is reflected by the DirtyDishesOnTableTrigger Event and is used as a way to trigger the “Waiter” that the customer is finished. In this example, the “Waiter” then responds by notifying the customer of what payment is due.

In Table 6-14, the DirtyDishesOnTableTrigger Event is defined by two characteristics: Table_identifier and Dishes_Dirty. The Table_identifier characteristic identifies the source responsible for the Event, which is the conceptual entity “Table,” and the Dishes_Dirty characteristic identifies the condition of the conceptual entity “Table” for triggering the Event. The conceptual entity “Waiter,” however, is not identified for this BOM Event as a target. That is because even though the “Waiter” might certainly have knowledge of the “Table” and designed to respond when there are “dirty dishes” on it, the “Table” and its “dirty dishes” have no knowledge of the “Waiter.” Thus, for a trigger, it is only important to associate the “Table” entity as the source and the characteristic Dishes_Dirty as the trigger condition for the BOM Event.

6.2.4.4.2 BOM Message Example

A message directed from one conceptual entity to another can be used as an event to fulfill an action within a pattern of interplay. For instance, in the restaurant payment pattern example provided in Table 6-14, a PaymentDueNotification message is directed from the “Waiter” to the “Customer.” In this example, the “Customer” then responds by paying the bill.

In Table 6-14, PaymentDueNotification Event is defined by the following characteristics: Waiter_identifier, Customer_identifier, and Amount. The Waiter_identifier characteristic identifies the source responsible for the Event, which is the conceptual entity “Waiter,” and the Customer_identifier characteristic identifies the target for the Event, which is the conceptual entity “Customer.” Finally, the Amount characteristic identifies the specific message content exchange from the source to the target.
6.3 Model Mapping

Model Mapping provides a mechanism for mapping between the elements of the Conceptual Model (see Section 6.2), and the interface description elements of the Object Model Definition, which are described using HLA OMT 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

Entity Type Mapping provides a mechanism for mapping between the entity type elements of the Conceptual Model and the interface description elements of the Object Model Definition, which are described using HLA OMT constructs.

6.3.1.2 Table Format

As depicted in Table 6-17, the Entity Type Mapping is intended to map Entity Types and their Characteristics with a supporting HLA Object/Interaction Class and their Attributes/Parameters.

<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 defined in the Entity Type Definition Table.</td>
<td>1..many</td>
<td>text</td>
</tr>
<tr>
<td>HLA Object/Interaction Class</td>
<td>Identifies an HLA object class or HLA interaction class defined in either the current BOM or in an external BOM that supports the specified Entity Type. If a class from an external BOM is being used, then a note reference should be made for the element. This supporting notes reference, which is defined in the Notes Table, should identify the iidtag of the supporting HLA Object/Interaction Class defined in the external BOM.</td>
<td>1..many</td>
<td>text</td>
</tr>
<tr>
<td>Characteristic</td>
<td>Identifies a characteristic of the entity type to be mapped.</td>
<td>0..many</td>
<td>text</td>
</tr>
<tr>
<td>HLA Attribute/Parameter</td>
<td>Identifies an HLA object class attribute or HLA interaction class parameter that supports the specified Entity Type Characteristic. An attribute or parameter can only be selected if its parent class has been identified as an HLA Object or Interaction class. If multiple HLA Object / Interaction Classes have been identified that can support the Entity Type, then the HLA object class or interaction class associated with the attribute or parameter that can support the identified Characteristic shall be identified using Dot Notation. If information from an external BOM is being used, then a note reference should be made for the element. This supporting notes reference, which is defined in the Notes Table, should identify the iidtag of the supporting HLA Attribute/Parameter defined in the external BOM.</td>
<td>1..many</td>
<td>text</td>
</tr>
<tr>
<td>Condition</td>
<td>Provides a free-text mechanism for identifying the conditions for use regarding each HLA Attribute/Parameter.</td>
<td>0..1</td>
<td>text</td>
</tr>
</tbody>
</table>

The Entity Type Mapping Table consists of a number of rows where each row represents a mapping of an Object Model Definition element to a Conceptual Model Entity Type element. As depicted in Table 6-18, Object Model Definition elements are described using HLA OMT constructs. These may include HLA Object Classes or HLA Interaction Classes, which are mapped to Entity Types, and the HLA Attributes or HLA Parameters associated with such an HLA OMT class, which are mapped to a Characteristic of an Entity Type.
Table 6-18 – Entity Type Mapping Table

<table>
<thead>
<tr>
<th>Category</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Entity Type</td>
</tr>
<tr>
<td>Entity Type</td>
<td>&lt;entity type name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- The first column (Category) specifies the categories of data that shall be provided in this table.
- The second column (Information) specifies the required information. Entries shall be required for all rows and cells except those where the information is shown in brackets. Unless otherwise stated in the definitions, each row shall appear only once.
  - The first Information column (Entity Type) is a reference to an Entity Type Definition defined in a row of the Entity Type Definition Table.
  - The second Information column (HLA Object/Interaction Class) identifies one or more HLA object classes or HLA interaction classes defined in the current BOM or an external BOM or FOM that support the Entity Type identified. If information from an external BOM is being used, then a note reference should be made for the element that identifies the idtag of the supporting HLA Object/Interaction Class defined in the external BOM.
  - The third Information column (Characteristic) contains references to characteristics of the entity type referenced in the first column. The characteristic must be defined in the Entity Type Definition Table. (Optional).
  - The fourth Information column (HLA Attribute/Parameter) identifies one or more HLA object class attributes or HLA interaction class parameters that support the Entity Type Characteristics identified. An attribute or parameter can only be selected if its parent class has been identified as an HLA Object or Interaction Class in the second column. If multiple HLA Object / Interaction Classes have been identified that can support the Entity Type, then the HLA object class or interaction class associated with the attribute or parameter that can support the identified Characteristic (third column) shall be identified in this column using Dot Notation. If information from an external BOM is being used, then a note reference should be made for the element that identifies the idtag of the supporting HLA Attribute/Parameter defined in the external BOM. (Conditional - if a Characteristic is defined, then an HLA Attribute/Parameter must also be defined).
  - The fifth Information column (Condition) provides a free-text mechanism for identifying the conditions for use regarding HLA Attributes/Parameters (Optional).

Lexicon Definitions and Notes, which are not reflected in this view, can also be identified for these elements.

A Dot Notation is to be used for the HLA values to ensure that the reference within the class structure is unambiguous. Thus, object class, interaction class, and their attribute and parameter names shall use Dot Notation as necessary, which may include the full parentage to uniquely identify the object model definition.
type being used to map with a conceptual model entity type. The actual object model definitions (see Section 6.4) may be contained in the present BOM for which the mapping has been defined, or may reference object model definitions captured within another BOM. If information from an external BOM is being used, then a note reference should be made for the HLA value element that identifies the idtag of the supporting HLA value that is defined in the external BOM.

6.3.1.3 Inclusion Criteria

Every BOM describing elements of a conceptual model may also contain a model mapping for an entity type. Likewise, every BOM describing interface elements of an object model may also contain a model mapping for an entity type. The categories of information specified in Table 6-18 shall be included for all BOMs, which contain an entity type mapping.

6.3.1.4 Example

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Entity Type</th>
<th>HLA Object/Interaction Class</th>
<th>Characteristics Ref</th>
<th>HLA Attributes/Parameters</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Table</td>
<td>HLAobjectRoot. PhysicalEntity.Table</td>
<td>Has Dirty Dishes</td>
<td>Table.hasDirtyDishes</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>Customer</td>
<td>HLAobjectRoot. PhysicalEntity.Human.Customer</td>
<td>Name</td>
<td>Customer.Id</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>Waiter</td>
<td>HLAobjectRoot. PhysicalEntity.Human.Waiter</td>
<td>Assigned Table</td>
<td>Customer.tableID</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>CashPayment</td>
<td>HLAobjectRoot. PhysicalEntity.Payment</td>
<td>Amount</td>
<td>Payment.Amount</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Customer ID</td>
<td>Human.Waiter.Customer.Id</td>
<td>na</td>
</tr>
</tbody>
</table>

A Dot Notation is used for the HLA values to ensure that the reference within the class structure is unambiguous. The actual object model definitions may be contained in the present BOM for which the mapping has been defined, or may reference object model definitions captured within another BOM. If information from an external BOM is being used, then a note reference should be made for the HLA value element that identifies the idtag of the supporting HLA value that is defined in the external BOM.

6.3.2 Event Type Mapping

6.3.2.1 Purpose/Background

Event Type Mapping provides a mechanism for mapping between the event type elements of the Conceptual Model and the interface description elements of the Object Model Definition, which are described using HLA OMT constructs.
6.3.2.2 Table Format

As depicted in Table 6-20, the Event Type Mapping is intended to map Event Types and their Characteristics with a supporting HLA Object/Interaction Class and their Attributes/Parameters.

<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 defined in the Event Type Definition Table.</td>
<td>1..many</td>
<td>text</td>
</tr>
<tr>
<td>HLA Object/Interaction Class</td>
<td>Identifies an HLA object class or HLA interaction class defined in either the current BOM or in an external BOM that supports the specified Event Type. If a class from an external BOM is being used, then a note reference should be made for the element. This supporting notes reference, which is defined in the Notes Table, should identify the idtag of the supporting HLA Object/Interaction Class defined in the external BOM.</td>
<td>1..many</td>
<td>text</td>
</tr>
<tr>
<td>Characteristic</td>
<td>Identifies a characteristic of the event type to be mapped. Characteristics include Source Characteristics, Target Characteristics, Content Characteristics, and/or Trigger Condition.</td>
<td>0..many</td>
<td>text</td>
</tr>
<tr>
<td>HLA Attribute/Parameter</td>
<td>Identifies an HLA object class attribute or HLA interaction class parameter that supports the specified Event Type Characteristic. An attribute or parameter can only be selected if its parent class has been identified as an HLA Object or Interaction class. If multiple HLA Object/Interaction Classes have been identified that support a specific Event Type, then the HLA object class or interaction class associated with the attribute or parameter that can support the identified Characteristic shall be identified using Dot Notation. If information from an external BOM is being used, then a note reference should be made for the element. This supporting notes reference, which is defined in the Notes Table, should identify the idtag of the supporting HLA Attribute/Parameter defined in the external BOM.</td>
<td>1..many</td>
<td>text</td>
</tr>
<tr>
<td>Condition</td>
<td>Provides a free-text mechanism for identifying the conditions for use regarding HLA Attributes/Parameters.</td>
<td>0..1</td>
<td>text</td>
</tr>
</tbody>
</table>

The Event Type Mapping Table consists of a number of rows where each row represents a mapping of an Object Model Definition element to a Conceptual Model Event Type element. As depicted in Table 6-21, mapped Object Model Definition elements are described using HLA OMT constructs. These may include HLA Object Classes or HLA Interaction Classes, which are mapped to Event Types, and the HLA Attributes or HLA Parameters associated with such an HLA OMT class, which are mapped to a Characteristic of an Event Type.
Table 6-21 – Event Type Mapping Table

<table>
<thead>
<tr>
<th>Category</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Type</td>
<td>&lt;event type ref&gt;</td>
</tr>
<tr>
<td>HLA Object/Interaction Classes</td>
<td>&lt;HLA class ref&gt;</td>
</tr>
<tr>
<td>Characteristic Ref</td>
<td>[&lt;characteristic ref&gt;]</td>
</tr>
<tr>
<td>HLA Attributes/Parameters</td>
<td>&lt;HLA property ref&gt;</td>
</tr>
<tr>
<td>Condition</td>
<td>[&lt;condition&gt;]</td>
</tr>
<tr>
<td>Event Type</td>
<td>&lt;event type ref&gt;</td>
</tr>
<tr>
<td>HLA Object/Interaction Classes</td>
<td>&lt;HLA class ref&gt;</td>
</tr>
<tr>
<td>Characteristic Ref</td>
<td>[&lt;characteristic ref&gt;]</td>
</tr>
<tr>
<td>HLA Attributes/Parameters</td>
<td>&lt;HLA property ref&gt;</td>
</tr>
<tr>
<td>Condition</td>
<td>[&lt;condition&gt;]</td>
</tr>
<tr>
<td>Event Type</td>
<td>&lt;event type ref&gt;</td>
</tr>
<tr>
<td>HLA Object/Interaction Classes</td>
<td>&lt;HLA class ref&gt;</td>
</tr>
<tr>
<td>Characteristic Ref</td>
<td>[&lt;characteristic ref&gt;]</td>
</tr>
<tr>
<td>HLA Attributes/Parameters</td>
<td>&lt;HLA property ref&gt;</td>
</tr>
<tr>
<td>Condition</td>
<td>[&lt;condition&gt;]</td>
</tr>
</tbody>
</table>

- The first column (Category) specifies the categories of data that shall be provided in this table.
- The second column (Information) specifies the required information. Entries shall be required for all rows except those where the information is shown in brackets. Unless otherwise stated in the definitions, each row shall appear only once.
  - The first Information column (Event Type Reference) is a reference to an event type definition defined in a row of the Event Type Definition Table.
  - The second Information column (HLA Object/Interaction Class) identifies one or more HLA object classes or HLA interaction classes in the current or an external BOM that support the Event Type identified. If information from an external BOM is being used, then a note reference should be made for the element that identifies the idtag of the supporting HLA Object/Interaction Class defined in the external BOM.
  - The third Information column (Characteristic Ref) contains references to characteristics of the event type referenced in the first column. The characteristic must be defined in the Event Type Definition Table. Characteristics include Source Characteristics, Target Characteristics, and Condition Characteristics. (Optional)
  - The fourth Information column (HLA Attributes/Parameters) identifies one or more HLA object class attributes or HLA interaction class parameters that support the Event Type Characteristics identified. An attribute or parameter can only be selected if its parent class has been identified as an HLA Object or Interaction class in the second column. If multiple HLA Object / Interaction Classes have been identified that can support the Event Type, then the HLA object class or interaction class associated with the attribute or parameter that can support the identified Characteristic (third column) shall be identified in this column using Dot Notation. If information from an external BOM is being used, then a note reference should be made for the element that identifies the idtag of the supporting HLA Attribute/Parameter defined in the external BOM. (Conditional - if a Characteristic is defined, then an HLA Attribute/Parameter must also be defined)
  - The fifth Information column (Condition) provides a free-text mechanism for identifying the conditions for use regarding HLA Attributes/Parameters (Optional)

Lexicon Definitions and Notes, which are not reflected in this view, can also be identified for these elements.

A Dot Notation is to be used for the HLA values to ensure that the reference within the class structure is unambiguous. Thus, object class, interaction class, and their attribute and parameter names shall use Dot Notation as necessary, which may include the full parentage to uniquely identify the object model definition.
type being used to map with a conceptual model event type. The actual object model definitions (see Section 6.4) may be contained in the present BOM for which the mapping has been defined, or may reference object model definitions captured within another BOM. If information from an external BOM is being used, then a note reference should be made for the HLA value element that identifies the idtag of the supporting HLA value that is defined in the external BOM.

### 6.3.2.3 Inclusion Criteria

Every BOM describing elements of a conceptual model may also contain a model mapping for an event type. Likewise, every BOM describing interface elements of an object model may also contain a model mapping for an event type. The categories of information specified in Table 6-21 shall be included for all BOMs, which contain an event type mapping.

### 6.3.2.4 Example

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

**Table 6-22 – Events Type Mapping Example**

<table>
<thead>
<tr>
<th>Category</th>
<th>Event Type</th>
<th>HLA Object/Interaction Classes</th>
<th>Characteristics</th>
<th>HLA Attributes/Parameters</th>
<th>Condition</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DirtyDishesOnTable</td>
<td>HLInteractionRoot, PhysicalEntity, Table</td>
<td>Table_Identifer</td>
<td>Table.tableId</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Dishes_Dirty</td>
<td>Table.hasDirtyDishes</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NonPayingCustomer</td>
<td>HLInteractionRoot, PhysicalEvent, CustomLeaves</td>
<td>Customer_Identifer</td>
<td>CustomerLeaves. customerId</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Customer_NotPaid</td>
<td>Customer.hasPaid</td>
<td>Trigger if Id == customerId and hasPaid == true</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CheckRequest</td>
<td>RealWorldMessage, RequestCheck</td>
<td>Customer_Identifer</td>
<td>customerId</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PaymentDue</td>
<td>PaymentDue</td>
<td>Waiter_Identifer</td>
<td>waiterId</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Customer_Identifer</td>
<td>customerId</td>
<td>na</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Amount</td>
<td>amount</td>
<td>na</td>
<td></td>
</tr>
</tbody>
</table>

A Dot Notation is used for the HLA values to ensure that the reference within the class structure is unambiguous. The actual object model definitions may be contained in the present BOM for which the mapping has been defined, or may reference object model definitions captured within another BOM. If information from an external BOM is being used, then a note reference should be made for the HLA value element that identifies the idtag of the supporting HLA value that is defined in the external BOM.
6.4 Object Model Definition

The Model Mapping section establishes the relationship of conceptual model elements with object model interface elements (characterized by using the HLA OMT constructs). Thus, a BOM shall contain, in this section, the portion of an object model definition necessary to describe the structure of the objects and interactions to be used in a federate or federation including their attributes, and parameters, which support the capabilities described in a BOM’s conceptual model. This object model definition shall be captured using the Object Class, Attribute, Interaction Class, Parameter, and Datatype Tables as defined in the OMT Specification.

Tables 6-23, 6-24, 6-25, 6-26 provide examples of the Object Model Definitions used to represent the Restaurant Payment BOM.

<table>
<thead>
<tr>
<th>Table 6-23 – HLA Object Classes Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
</tr>
<tr>
<td>Object Class</td>
</tr>
<tr>
<td>PhysicalEntity</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>AbstractEntity</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 6-24 – HLA Attributes Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
</tr>
<tr>
<td>Object Class</td>
</tr>
<tr>
<td>HLAObjectRoot</td>
</tr>
<tr>
<td>PhysicalEntity, Table</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>PhysicalEntity, Customer</td>
</tr>
<tr>
<td>AbstractEntity, CashPayment</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 6-25 – HLA Interaction Classes Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
</tr>
<tr>
<td>Interaction Class</td>
</tr>
<tr>
<td>Physical Event</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>RealWorldMessage</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Copyright © 2004-2005 SISO. All rights reserved.
This is an unapproved SISO Draft Specification, subject to change.
Table 6-26 – HLA Parameters Example

<table>
<thead>
<tr>
<th>Category</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction Class Ref</td>
<td>Parameter</td>
</tr>
<tr>
<td>PhysicalEvent</td>
<td>customerID</td>
</tr>
<tr>
<td>PhysicalEvent. PresentCreditCard</td>
<td>cardNumber</td>
</tr>
<tr>
<td>HLAreinteractionRoot. PhysicalEvent</td>
<td>handful ID</td>
</tr>
<tr>
<td>RealWorldMessage. PaymentDue</td>
<td>amount</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Datatype</th>
<th>Available Dimensions</th>
<th>Transportation</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhysicalEvent</td>
<td>ID</td>
<td>NA</td>
<td>HLAreliable</td>
<td>Receive</td>
</tr>
<tr>
<td>PhysicalEvent. PresentCreditCard</td>
<td>String</td>
<td>NA</td>
<td>HLAreliable</td>
<td>Receive</td>
</tr>
<tr>
<td>HLAreinteractionRoot. PhysicalEvent</td>
<td>handful ID</td>
<td>NA</td>
<td>HLAreliable</td>
<td>Receive</td>
</tr>
<tr>
<td>RealWorldMessage. PaymentDue</td>
<td>Float32BE</td>
<td>NA</td>
<td>HLAreliable</td>
<td>Receive</td>
</tr>
</tbody>
</table>

### 6.5 Supporting 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. The BOM Pattern Description, Triggers, Messages, and State Machine Description Tables have explicit column and row entries for notes as can be seen in the examples for Tables 6-7, 6-8, 6-9, and 6-11. This notes feature permits users to associate explanatory information with individual tables and sub tables to facilitate effective use of the data. Following the convention of the 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 OMT Specification (Section 4.13.2 – Table 35).

##### 6.5.1.3 Inclusion Criteria

BOMs describing object classes, patterns of interplay, and/or state machines may include notes wherever such annotation improves the clarity and understandability of the conceptual model, model mapping or object model definition.

##### 6.5.1.4 Example

Table 6-27 provides an example of the use of the notes feature. Tables 6-7, 6-10, and 6-16 provide examples containing references to the notes in this table.
Table 6-27 – Notes Table Example

<table>
<thead>
<tr>
<th>Category</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notes</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>This pattern might be able to be used as a template to support payment patterns for other service oriented businesses</td>
</tr>
<tr>
<td>2</td>
<td>This pattern action may involve other employees including the maître d’ (i.e. Greeter)</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>
</tr>
<tr>
<td>4</td>
<td>Customer may also request check from other employee conceptual entity types</td>
</tr>
<tr>
<td>5</td>
<td>This is the idle state for the Greeter or Waiter</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. Any semantic descriptions that have been reflected in the various table views previously are contained separately within Lexicon Tables described in this section. By maintaining separate Lexicon Tables, the BOM lexicon provides a means for simulation software designers to define all entity types, event types, entity type characteristics, event type characteristics, pattern actions, state machine states in BOMs. For object classes, interaction classes, object class attributes, and interaction parameters, the Lexicon Tables provided in OMT Specification shall be used.

6.5.2.2 Pattern Description Definition Table

This subclause provides the format for describing pattern descriptions. The template that shall be used for this information is provided in Table 6-28.

Table 6-28 – BOM Pattern Description Definition Table

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;pattern&gt;</td>
<td>&lt;definition&gt;</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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

Table 6-29 provides an example of the use of the table. The pattern is taken from the examples provided earlier in Section 6.

Table 6-29 – BOM Pattern Description Definition Table Example

<table>
<thead>
<tr>
<th>Pattern</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>
</tbody>
</table>
6.5.2.3 Pattern Action Definition Table

This subclause provides the format for describing pattern action, exception, or variation descriptions. The template that shall be used for this information is provided in Table 6-30.

Table 6-30 – BOM Pattern Action Definition Table

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

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

Table 6-31 provides an example of the use of the table. The pattern actions are taken from the examples provided earlier in Section 6.

Table 6-31 – BOM Pattern Action Definition Table Example

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Action</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RestaurantPayment</td>
<td>CustomerDone</td>
<td>Waiter is made aware that the 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>
<tr>
<td>RestaurantPayment</td>
<td>CustomerPays.BillPaid_Cash</td>
<td>The customer pays by cash</td>
</tr>
<tr>
<td>RestaurantPayment</td>
<td>CustomerPays.BillPaid_Credit</td>
<td>The customer pays by credit card</td>
</tr>
<tr>
<td>RestaurantPayment</td>
<td>CustomerPays.BillNotPaid</td>
<td>The customer leaves without paying</td>
</tr>
</tbody>
</table>

6.5.2.4 State Machine Description Definition Table

This subclause provides the format for describing pattern descriptions. The template that shall be used for this information is provided in Table 6-32.

Table 6-32 – BOM State Machine Description Definition Table

<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>
</tbody>
</table>
The first column (State Machine) shall contain the names of all state machines described in the BOM.

The second column (Definition) shall describe the semantics for the state machine.

Table 6-33 provides an example of the use of the table. The state machines are taken from the examples provided earlier in Section 6.

Table 6-33 – 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 subclause provides the format for describing pattern descriptions. The template that shall be used for this information is provided in Table 6-34.

Table 6-34 – BOM State Machine States Definition Table

<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;action&gt;</td>
<td>&lt;definition&gt;</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>&lt;state machine&gt;</td>
<td>&lt;action&gt;</td>
<td>&lt;definition&gt;</td>
</tr>
</tbody>
</table>

The first column (State Machine) shall contain the names of all 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 6-35 provides an example of the use of the table. The patterns are taken from the examples provided earlier in Section 6.

Table 6-35 – BOM State Machine States Definition Table Example

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

This subclause provides the format for describing entity types. The template that shall be used for this information is provided in Table 6-36.

Table 6-36 – BOM Entity Type Definition Table

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

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

Table 6-37 provides an example of the use of the table. The entity types are taken from the examples provided earlier in Section 6.

Table 6-37 – BOM Entity Type Definition Table Example

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place</td>
<td>where a meal is served</td>
</tr>
<tr>
<td>Table</td>
<td>Customer</td>
</tr>
<tr>
<td></td>
<td>Customer paying for a meal</td>
</tr>
<tr>
<td>Waiter</td>
<td>Waiter which served the meal and processes payment</td>
</tr>
<tr>
<td>CashPayment</td>
<td>Money exchanged for meal</td>
</tr>
</tbody>
</table>

6.5.2.7 Entity Type Characteristic Definition Table

This subclause provides the format for describing entity type characteristics. The template that shall be used for this information is provided in Table 6-38.

Table 6-38 – BOM Entity Type Characteristic Definition Table

<table>
<thead>
<tr>
<th>Type</th>
<th>Characteristic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;entity type&gt;</td>
<td>&lt;characteristic&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;characteristic&gt;</td>
<td>&lt;definition&gt;</td>
</tr>
</tbody>
</table>

- The first column (Entity Type) shall contain the names of all 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 6-39 provides an example of the use of the table. The entity type characteristics are taken from the examples provided earlier in Section 6.

<table>
<thead>
<tr>
<th>Type</th>
<th>Characteristic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Has Dirty Dishes</td>
<td>fact that the table has dirty dishes</td>
</tr>
<tr>
<td>Customer</td>
<td>Name</td>
<td>a customer identifier</td>
</tr>
<tr>
<td>Waiter</td>
<td>Assigned Tables</td>
<td>Table the customer is at</td>
</tr>
<tr>
<td>CashPayment</td>
<td>Amount</td>
<td>Amount of money</td>
</tr>
<tr>
<td></td>
<td>Waiter ID</td>
<td>The waiter processing payment</td>
</tr>
<tr>
<td></td>
<td>Customer ID</td>
<td>The customer giving payment</td>
</tr>
</tbody>
</table>

6.5.2.8 Event Type Definition Table

This subclause provides the format for describing entity types. The template that shall be used for this information is provided in Table 6-40.

<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>
<tr>
<td>...</td>
<td>...</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 6-41 provides an example of the use of the table. The event types are taken from the examples provided earlier in Section 6.

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirtyDishesOnTable</td>
<td>There is 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 subclause provides the format for describing 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 6-42.
Table 6-42 – BOM Event Type Characteristic Definition Table

<table>
<thead>
<tr>
<th>Type</th>
<th>Characteristic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;event type&gt;</td>
<td>&lt;characteristic&gt;</td>
<td>&lt;definition&gt;</td>
</tr>
<tr>
<td>&lt;event type&gt;</td>
<td>&lt;characteristic&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 6-43 provides an example of the use of the table. The event types are taken from the examples provided earlier in Section 6.

Table 6-43 – 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>Table_identifier</td>
<td>Table identifier</td>
</tr>
<tr>
<td></td>
<td>Dishes_Dirty</td>
<td>that fact the table has dirty dishes</td>
</tr>
<tr>
<td></td>
<td>Customer_identifier</td>
<td>a customer identifier</td>
</tr>
<tr>
<td></td>
<td>Customer_NotPaid</td>
<td>The customer who left has not paid</td>
</tr>
<tr>
<td>CheckRequest</td>
<td>Customer_Identifier</td>
<td>The person making the request</td>
</tr>
<tr>
<td></td>
<td>Waiter_Identifier</td>
<td>The waiter to whom the request is directed</td>
</tr>
<tr>
<td>PaymentDueNotification</td>
<td>Waiter_Identifier</td>
<td>a customer identifier</td>
</tr>
<tr>
<td></td>
<td>Customer_Identifier</td>
<td>The customer owing money</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 the Extensible Markup Language (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.

Figure 7-1 – BOM DIF Schema Hierarchy
The major elements of the BOM schema are identified as:

- **modelIdentification**
  - poc
  - keyword
  - reference
  - glyph

- **conceptualModel**
  - patternDescription
  - stateMachine
  - entityType
  - eventType

- **modelMapping**
  - entityTypeMapping
  - eventTypeMapping

- **objectModelDefinition**
  - objectClasses
  - objectClassAttributes
  - interactionClasses
  - interactionClassParameters
  - data types

- **notes**

The attribute nodes identified in Figure 7-1 and depicted 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 element.

**Figure 7-2 – BOM DIF Schema Common Attributes**

Support for the BOM lexicon is provided within each of the elements as described in Section 6 using a semantics sub-element. The remainder of this section describes the individual schema components for each major element.

### 7.1 Model Identification

The modelIdentification provides a mechanism for identifying the essential metadata of a BOM. The structure for the modelIdentification data type for this metadata model is illustrated in Figure 7-3.
The keyword element identified in the `modelIdentification` template component is represented as a complex data type as depicted in Figure 7-4.
The POC element identified in the `modelIdentification` template component is represented as a complex data type as depicted in Figure 7-5.

The reference element identified in the `modelIdentification` template component is represented as a complex data type as depicted in Figure 7-6.

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.
he patternDescription provides a mechanism for identifying actions (including variations and exceptions) necessary for fulfilling the pattern of interplay that is being represented by the BOM. It is these actions, variations and exceptions that tie in with a defined event within the BOM, or reference a completely unique BOM. As illustrated in Figure 7-8, the pattern data type is intended to allow these actions, exceptions and variations to be identified.

**Figure 7-8 – BOM pattern Element**
Notice that the pattern describing a BOM can be represented by one or more actions. Each action can also include an exception, which is an unexpected action that might occur, and a variation, which identifies a different way for the action to be accomplished. The action data type element identified in the pattern template component is represented as a complex data type as depicted in Figure 7-9.

The exception data type and variation data type element identified in the pattern template component, which is used for representing an exception and variation of an action, are represented as complex data types as depicted in Figure 7-10 and 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 or variation element.

7.2.2 State Machine

The stateMachine data type provides a mechanism for identifying states of a conceptual entity to support one or more patterns of interplay. The components used to represent a state machine 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 Description (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 entityType provides a mechanism for identifying conceptual entities. The components used to represent entityTypes are depicted in Figure 7-13.
7.2.4 BOM Event Types

The `eventType` provides a mechanism for identifying conceptual events. The components used to represent `eventTypes` are depicted in Figure 7-14.
One or more triggers or messages can be chosen for representing the events pertaining to the pattern a BOM is intended to characterize. A trigger represents an undirected simulation event without any information about intended receiver(s). A message represents a directed simulation event including information on intended receiver.

7.3 Model Mapping

The `modelMapping` data type provides a mechanism for identifying the mapping between `entity` and `event` elements of the conceptual model with HLA OMT interface elements of the object model definition. The elements of the conceptual model and the interface elements of the Object Model Definition that are being mapped may be defined either within the current BOM or an external BOM. The XML design for representing the template components used to support the `modelMapping` of a BOM is depicted in Figure 7-15.

![Figure 7-15 – BOM mappingType Segment](image)

The `characteristic` element identified in Figure 7-15 for the `mappingType` is supported using the complex type depicted in Figure 7-16.

![Figure 7-16 – BOM characteristicMappingType Segment](image)
7.4 Object Model Definition

The objectModelDefinition data type provides a mechanism for identifying the interface elements necessary to support the capabilities described in the conceptualModel of a BOM in the context of HLA Object Classes and Interaction Classes. An excerpt of the HLA OMT Specification is used to identify what is needed for representing the objectModelDefinition of a BOM. The HLA OMT components used to support the objectModelDefinition of a BOM are depicted in Figure 7-17.

Figure 7-17– BOM modelDefinition Segment

What is not leveraged from the original HLA OMT Specification are the federate/federation connections related to technical interoperability. This has been considered a deeper level of detail not needed for representing simulation patterns and components.

Copyright © 2004-2005 SISO. All rights reserved.
This is an unapproved SISO Draft Specification, subject to change.
The HLA OMT data types of interest for a BOM include the following:

- **Object classes / attributes**
- **Interaction classes / parameters**
- **dataTypes**, which includes
  - **basicDataRepresentations**
  - **simpleDataTypes**
  - **enumeratedDataTypes**
  - **arrayDataTypes**
  - **fixedRecordDataTypes**
  - **variantRecordDataTypes**

Those not implicitly used include:

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

### 7.5 Notes

The *notes* data type, which is leveraged from the OMT Specification, provides a mechanism for associating notes with any of the BOM template components. Within the BOM, any XML Element can be supported by a notes reference. The actual note to the notes reference would be contained in the Notes Table. The components used to represent notes within the Notes Table are depicted in Figure 7-18.
This page left intentionally blank
8 Annex A – BOM Schemas

The listings of the BOM related-schemas are provided below.

Model Identification (ModelID) Namespace

The Model Identification (ModelID) namespace is defined within a separate schema, identified as `modelid_v0_4.xsd`, and is used by the BOM to represent metadata. 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 containing this `modelID` namespace.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by Tram Chase (SimVentions, Inc.) -->
  <xs:complexType name="String">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attributeGroup ref="commonAttributes"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="IdentifierType">
    <xs:simpleContent>
      <xs:extension base="xs:NCName">
        <xs:attributeGroup ref="commonAttributes"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="NonEmptyString">
    <xs:simpleContent>
      <xs:extension base="nonEmptyString">
        <xs:attributeGroup ref="commonAttributes"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:simpleType name="OMTypeEnumerations">
    <xs:restriction base="xs:string">
      <xs:enumeration value="FOM"/>
      <xs:enumeration value="SOM"/>
      <xs:enumeration value="BOM"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="OMTypeUnion">
    <xs:union memberTypes="OMTypeEnumerations nonEmptyString"/>
  </xs:simpleType>
  <xs:simpleType name="glyphTypeEnumerations">
    <xs:restriction base="xs:string">
      <xs:enumeration value="BITMAP"/>
      <xs:enumeration value="JPG"/>
      <xs:enumeration value="GIF"/>
      <xs:enumeration value="PNG"/>
      <xs:enumeration value="TIFF"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="glyphTypeUnion">
    <xs:union memberTypes="glyphTypeEnumerations xs:string"/>
  </xs:simpleType>
  <xs:complexType name="glyphType" mixed="true">
    <xs:simpleContent>
      <xs:extension base="xs:base64Binary">
        <xs:attributeGroup ref="commonAttributes"/>
        <xs:attribute name="type" type="glyphTypeUnion"/>
        <xs:attribute name="height" type="xs:short"/>
        <xs:attribute name="width" type="xs:short"/>
        <xs:attribute name="alt" type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="pocType">
  </xs:complexType>
</xs:schema>
```
<xs:sequence>
  <xs:element name="pocType" type="poctypeType"/>
</xs:sequence>

<xs:element>
  <xs:annotation>
    <xs:documentation>specifies the role that the POC has with respect to the model</xs:documentation>
  </xs:annotation>
  <xs:element>
    <xs:element name="pocName" type="String" minOccurs="0"/>
    <xs:element name="pocOrg" type="String" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="pocTelephone" type="String" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="pocEmail" type="String" maxOccurs="unbounded"/>
  </xs:element>
</xs:element>

</xs:complexType>

<xs:attributeGroup name="commonAttributes">
  <!--this is the common attributes for any element -->
  <xs:attribute name="notes" type="xs:IDREFS" use="optional"/>
  <xs:attribute name="idtag" type="xs:ID" use="optional"/>
  <xs:anyAttribute namespace="##other"/>
</xs:attributeGroup>

<xs:simpleType name="nonEmptyString">
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="SecurityClassificationEnumeration">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Unclassified"/>
    <xs:enumeration value="Confidential"/>
    <xs:enumeration value="Secret"/>
    <xs:enumeration value="Top Secret"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="SecurityClassificationUnion">
  <xs:union memberTypes="SecurityClassificationEnumeration nonEmptyString"/>
</xs:simpleType>

<xs:complexType name="modelType">
  <xs:simpleContent>
    <xs:extension base="OMTypeUnion">
      <xs:attributeGroup ref="commonAttributes"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="securityClassificationType">
  <xs:simpleContent>
    <xs:extension base="SecurityClassificationUnion">
      <xs:attributeGroup ref="commonAttributes"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:simpleType name="ApplicationDomainEnumerations">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Analysis"/>
    <xs:enumeration value="Training"/>
    <xs:enumeration value="Test and Evaluation"/>
    <xs:enumeration value="Engineering"/>
    <xs:enumeration value="Acquisition"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ApplicationDomainUnion">
  <xs:union memberTypes="ApplicationDomainEnumerations xs:string"/>
</xs:simpleType>

<xs:complexType name="applicationDomainType">
  <xs:simpleContent>
    <xs:extension base="ApplicationDomainUnion">
      <xs:attributeGroup ref="commonAttributes"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:simpleType name="POCTypeEnumeration">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Primary author"/>
    <xs:enumeration value="Contributor"/>
    <xs:enumeration value="Proponent"/>
    <xs:enumeration value="Sponsor"/>
    <xs:enumeration value="Release authority"/>
    <xs:enumeration value="Technical POC"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="POCTypeUnion">
</xs:simpleType>
<xs:union memberTypes="POCTypeEnumeration nonEmptyString"/>
<xs:complexType name="pocTypeType">
  <xs:simpleContent>
    <xs:extension base="POCTypeUnion">
      <xs:attributeGroup ref="commonAttributes"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:simpleType name="referenceTypeEnumerations">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Source Material"/>
    <xs:enumeration value="Conceptual Model"/>
    <xs:enumeration value="Related BOM"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="referenceTypeUnion">
  <xs:union memberTypes="referenceTypeEnumerations xs:string"/>
  <!--this allows enumerations to be used plus any user defined content -->
</xs:simpleType>

<xs:complexType name="referenceType">
  <xs:sequence>
    <xs:element name="type">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="referenceTypeUnion">
            <xs:attributeGroup ref="commonAttributes"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="identification">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:anyURI">
            <xs:attributeGroup ref="commonAttributes"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attributeGroup ref="commonAttributes"/>
</xs:complexType>

<xs:complexType name="keywordType">
  <xs:sequence>
    <xs:element name="taxonomy" type="String" minOccurs="0"/>
    <xs:element name="keywordValue" type="NonEmptyString"/>
  </xs:sequence>
  <xs:attributeGroup ref="commonAttributes"/>
</xs:complexType>

<xs:complexType name="modelIdentificationType">
  <xs:sequence>
    <xs:element name="name" type="IdentifierType">
      <xs:annotation>
        <xs:documentation>specifies the name assigned to the object model</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="type" type="modelType">
      <xs:annotation>
        <xs:documentation>specify the type of model that is represented</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="version" type="NonEmptyString">
      <xs:annotation>
        <xs:documentation>specifies the version identification assigned to the object model</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="modificationDate" nillable="false">
      <xs:annotation>
        <xs:documentation>specifies the latest date on which this version of the object model was created or modified. The modification date shall be specified in the format "YYYY-MM-DD"</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attributeGroup ref="commonAttributes"/>
</xs:complexType>
BOM Schema

The following listing provides the core XML schema representing the BOM DIF. This listing of the BOM schema, identified as BOM_v0_12.xsd, imports the modelID namespace and schema for representing metadata, and uses the HLA OMT namespace and schema for representing the interface elements of the Object Model Definition.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by Tram Chase (SimVentions, Inc.) -->
<!-- created with XML Spy v4.2 U (http://www.xmlspy.com) by Björn Löfstrand (Pitch Kunskapsutveckling AB) -->
<!--W3C Schema developed by the BOM Product Develop Group (PDG) - version 0.12 -->
<!--Schema for BOM DIF

Version 0.11 - June 29, 2005
Version 0.12 - Sept 8, 2005

targetNamespace="http://www.sisostds.org/schemas/bom" elementFormDefault="qualified">
  <xs:import namespace="http://www.sisostds.org/schemas/IEEE1516.2-2006" schemaLocation="IEEE1516.2-2006-D2v0.81.xsd"/>
  <xs:import namespace="http://www.sisostds.org/schemas/modelID" schemaLocation="modelID_v0_4.xsd"/>
  <xs:annotation>
    <xs:documentation xml:lang="en">Schema for BOM DIF</xs:documentation>
  </xs:annotation>
  <xs:element name="BOM" type="BOMType"/>
  <xs:simpleType name="blName">
    <xs:restriction base="xs:string">
      <xs:pattern value="\S*"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="patternType" id="patternDescription">
    <xs:sequence>
      <xs:element name="name" type="modelID:IdentifierType"/>
      <xs:element name="action" maxOccurs="unbounded">
        <xs:complexType>
          <xs:complexContent>
            <xs:extension base="actionType">
              <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element name="exception" type="exceptionType"/>
                <xs:element name="variation" type="variationType"/>
              </xs:choice>
            </xs:extension>
          </xs:complexContent>
        </xs:complexType>
      </xs:element>
      <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
    </xs:sequence>
    <xs:attributeGroup ref="modelID:commonAttributes"/>
  </xs:complexType>
  <xs:complexType name="actionType">
    <xs:sequence>
      <xs:element name="sequence" type="blName"/>
    </xs:sequence>
    <xs:attributeGroup ref="modelID:commonAttributes"/>
  </xs:complexType>
  <xs:complexType name="bom" minOccurs="0"/>
    <xs:sequence>
      <xs:element name="sequence" type="bom"/>
    </xs:sequence>
    <xs:attributeGroup ref="modelID:commonAttributes"/>
  </xs:complexType>
</xs:schema>
```
<xs:complexType name="entityType">
  <xs:sequence>
    <xs:element name="name" type="modelID:IdentifierType"/>
    <xs:element name="characteristic" type="characteristicType" maxOccurs="unbounded"/>
    <xs:element name="semantics" type="modelID:String" minOccurs="0">
      <xs:annotation>
        <xs:documentation>lexicon entry for this entity type</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="eventType">
  <xs:sequence>
    <xs:element name="name" type="modelID:IdentifierType"/>
    <xs:element name="sourceCharacteristic">
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="characteristicType"/>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="targetCharacteristic" type="characteristicType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="contentCharacteristic" type="characteristicType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="triggerCondition" type="modelID:NonEmptyString" minOccurs="0"/>
    <xs:element name="semantics" type="modelID:String" minOccurs="0">
      <xs:annotation>
        <xs:documentation>lexicon entry for this event type</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="mappingType">
  <xs:sequence>
    <xs:element name="mappedType" type="mappedTypeType" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="mappedTypeType">
  <xs:sequence>
    <xs:element name="name" type="modelID:IdentifierType"/>
    <xs:element name="hlaClass" type="modelID:NonEmptyString" maxOccurs="unbounded"/>
    <xs:element name="characteristic" type="characteristicMappingType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="BOMType">
  <xs:sequence>
    <xs:element name="modelIdentification" type="modelID:modelIdentificationType"/>
    <xs:element name="conceptualModel" type="conceptualModelType" minOccurs="0"/>
    <xs:element name="modelMapping" type="modelMappingType" minOccurs="0"/>
    <xs:element name="objectModelDefinition" type="objectModelDefinitionType" minOccurs="0"/>
    <xs:element name="notes" minOccurs="0">
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="omt:notesType"/>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="modelMappingType">
  <xs:sequence>
    <xs:element name="entityTypeMappings" minOccurs="0">
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="mappingType"/>  
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="eventTypeMappings" minOccurs="0">
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="mappingType"/>  
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
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. This listing of the HLA OMT schema, identified as IEEE1516.2-2006-D2v0.81.xsd, also imports the modelID namespace and schema for representing metadata.

```xml
<?xml version='1.0' encoding='UTF-8'?>
<!- modified with XMLSpy v2005 rel. 3 U (http://www.altova.com) by Tram Chase (SimVentions, Inc.) -->
<!- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by Bjorn Lofstrand (Pitch Technologies) -->
  <xs:import namespace='http://www.sisostds.org/schemas/modelID' schemaLocation='modelID_v0_4.xsd'/>
  <xs:element name='objectModel' type='objectModelType'>
    <xs:key name='dimensionDatatypeKey'>
      <xs:selector xpath='.//ns:simpleData/ns:name | .//ns:enumeratedData/ns:name'/>
      <xs:field xpath='.'/>
    </xs:key>
    <xs:keyref name='dimensionDatatypeRef' refer='dimensionDatatypeKey'/>
    <xs:selector xpath='./ns:dimensions/*/ns:dataType'/>
    <xs:field xpath='.'/>
  </xs:element>
  <xs:element name='objectClass'>
    <xs:complexType>
      <xs:complexContent>
        <xs:extension base='objectClassType'/>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Copyright © 2004-2005 SISO. All rights reserved. This is an unapproved SISO Draft Specification, subject to change.
<xs:documentation>ensures uniqueness of objectClass names among class siblings</xs:documentation>

<xs:selector xpath="./ns:objectClass"/>
<xs:field xpath="ns:name"/>
</xs:unique>

<xs:unique name="attributeName">
<xs:selector xpath="./ns:attribute"/>
<xs:field xpath="ns:name"/>
</xs:unique>

<xs:element name="interactionClass">
<xs:complexType>
<xs:complexContent>
<xs:extension base="interactionClassType"/>
</xs:complexContent>
</xs:complexType>
</xs:element>

<xs:complexType name="switchType">
<xs:simpleContent>
<xs:extension base="enabled_disabledEnumerations">
<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>

<xs:complexType name="tagType">
<xs:sequence>
<xs:element name="dataType" type="ReferenceType">
<xs:annotation>
<xs:documentation>identifies the datatype for the user-defined tag</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element name="semantics" type="modelID:String" minOccurs="0">
<xs:annotation>
<xs:documentation>expands and describes the use of the datatype for the user-supplied tag</xs:documentation>
</xs:annotation>
</xs:element>
<xs:any namespace="##other" minOccurs="0"/>
</xs:sequence>
<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="objectsType">
<xs:sequence>
<xs:element ref="objectClass"/>
<xs:any namespace="##other" minOccurs="0"/>
</xs:sequence>
<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="interactionsType">
<xs:sequence>
<xs:element ref="interactionClass"/>
<xs:any namespace="##other" minOccurs="0"/>
</xs:sequence>
<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:documentation>identifies the datatype of the parameter</xs:documentation>

<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 name="sharing" type="sharingType" default="Neither">
  <xs:annotation>
    <xs:documentation>specifies publication and subscription capabilities of this object</xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element name="semantics" type="modelID:String" minOccurs="0">
  <xs:annotation>
    <xs:documentation>lexicon entry for this object class</xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element ref="attribute" minOccurs="0" maxOccurs="unbounded"/>

<xs:element ref="objectClass" minOccurs="0" maxOccurs="unbounded"/>

<xs:any namespace="##other" minOccurs="0"/>

<xs:attributeGroup ref="modelID:commonAttributes"/>

<xs:complexType name="objectClassType">
  <xs:sequence>
    <xs:element name="name" type="modelID:IdentifierType"/>
    <xs:element name="sharing" type="sharingType" default="Neither">
      <xs:annotation>
        <xs:documentation>specifies publication and subscription capabilities of this object class</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="semantics" type="modelID:String" minOccurs="0">
      <xs:annotation>
        <xs:documentation>lexicon entry for this object class</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element ref="attribute" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="objectClass" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="interactionClassType">
  <xs:sequence>
    <xs:element name="name" type="modelID:IdentifierType"/>
    <xs:element name="sharing" type="sharingType">
      <xs:annotation>
        <xs:documentation>specifies publication and subscription capabilities of this interaction class</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="dimensions" minOccurs="0">
      <xs:annotation>
        <xs:documentation>records the association of the interaction class with a set of dimensions if a federate or federation is using DDM services for this attribute</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="dimension" type="ReferenceType" minOccurs="0" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>identifies a dimension associated with this interaction class</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:any namespace="##other" minOccurs="0"/>
      </xs:sequence>
      <xs:attributeGroup ref="modelID:commonAttributes"/>
    </xs:complexType>
  </xs:sequence>
  <xs:element name="transportation" type="ReferenceType">
    <xs: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:element name="semantics" type="modelID:String" minOccurs="0">
    <xs:annotation>
      <xs:documentation>lexicon entry for this interaction class</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element ref="attribute" minOccurs="0" maxOccurs="unbounded"/>
  <xs:element ref="interactionClass" minOccurs="0" maxOccurs="unbounded"/>
</xs:complexType>
<xs:complexType name="dimensionsType">
  <xs:sequence>
    <xs:element name="dimension" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="dimensionType"/>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="lookahead" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="dimensionType">
  <xs:sequence>
    <xs:element name="name" type="modelID:IdentifierType"/>
    <xs:element name="dataType" type="ReferenceType">
      <xs:annotation>
        <xs:documentation>identifies the datatype for the federate view of the dimension</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="upperBound" minOccurs="0">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:positiveInteger">
            <xs:attributeGroup ref="modelID:commonAttributes"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="normalization" type="modelID:NonEmptyString">
      <xs:annotation>
        <xs:documentation>specifies the map from a subscription/update region's bounding 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:any namespace="##other" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="timeType">
  <xs:sequence>
    <xs:element name="timeStamp" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="dataType" type="ReferenceType">
            <xs:annotation>
              <xs:documentation>identifies the timestamp datatype</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element name="semantics" type="modelID:String" minOccurs="0">
            <xs:annotation>
              <xs:documentation>expands and describes the use of the timestamp datatype for timestamp</xs:documentation>
            </xs:annotation>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:any namespace="##other" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

Copyright © 2004-2005 SISO. All rights reserved.
This is an unapproved SISO Draft Specification, subject to change.
<xs:complexType>
  <xs:sequence>
    <xs:element name="dataType" type="ReferenceType">
      <xs:annotation>
        <xs:documentation>identifies the lookahead datatype</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="semantics" type="modelID:String" minOccurs="0">
      <xs:annotation>
        <xs:documentation>expands and describes the use of the datatype for lookahead</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="tagsType">
  <xs:sequence>
    <xs:element name="updateReflectTag" type="tagType" minOccurs="0"/>
    <xs:element name="sendReceiveTag" type="tagType" minOccurs="0"/>
    <xs:element name="deleteRemoveTag" type="tagType" minOccurs="0"/>
    <xs:element name="divestitureRequestTag" type="tagType" minOccurs="0"/>
    <xs:element name="divestitureCompletionTag" type="tagType" minOccurs="0"/>
    <xs:element name="acquisitionRequestTag" type="tagType" minOccurs="0"/>
    <xs:element name="requestUpdateTag" type="tagType" minOccurs="0"/>
    <xs:any namespace="##other" minOccurs="0"/>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="synchronizationPointType">
  <xs:sequence>
    <xs:element name="label" type="modelID:IdentifierType"/>
    <xs:element name="dataType" type="ReferenceType" minOccurs="0">
      <xs:annotation>
        <xs:documentation>identifies the datatype of the user-supplied tag for those synchronization points that the federate or federation designate as providing a user-supplied tag</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="capability" type="capabilityType">
      <xs:annotation>
        <xs:documentation>indicates the level of interaction that a federate is capable of honoring</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="synchronizationsType">
  <xs:sequence>
    <xs:element name="synchronizationPoint" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="synchronizationPointType"/>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
    <xs:any namespace="##other" minOccurs="0"/>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="transportationsType">
  <xs:sequence>
    <xs:element name="transportation" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="transportationPointType"/>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
    <xs:any namespace="##other" minOccurs="0"/>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
<xs:extension base="transportationType"/>
</xs:complexType>
</xs:element>
</xs:complexContent>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:element>
<xs:element name="resolution" type="modelID:NonEmptyString" minOccurs="0">
  <xs:annotation>
    <xs:documentation>describes the precision of measure for the datatype</xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element name="accuracy" type="modelID:NonEmptyString" minOccurs="0">
  <xs:annotation>
    <xs:documentation>describes maximum deviation of the value from its intended value in the federate or federation</xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element name="semantics" type="modelID:String" minOccurs="0"/>
<xs:any namespace="##other" minOccurs="0"/>
</xs:sequence>
<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="fixedRecordDataType">
  <xs:sequence>
    <xs:element name="name" type="modelID:IdentifierType"/>
    <xs:element name="encoding" type="fixedRecordEncodingType">
      <xs:annotation>
        <xs:documentation>describes the encoding of the fixed record datatype (i.e., the organization of fields) as delivered to and received from the RTI</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="field" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="name" type="modelID:IdentifierType"/>
          <xs:element name="dataType" type="ReferenceType"/>
          <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
          <xs:any namespace="##other" minOccurs="0"/>
        </xs:sequence>
        <xs:attributeGroup ref="modelID:commonAttributes"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
    <xs:element name="field" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="name" type="modelID:IdentifierType"/>
          <xs:element name="dataType" type="ReferenceType"/>
          <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
          <xs:any namespace="##other" minOccurs="0"/>
        </xs:sequence>
        <xs:attributeGroup ref="modelID:commonAttributes"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="enumerator" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="name" type="modelID:IdentifierType"/>
          <xs:element name="value" type="modelID:String" maxOccurs="unbounded">
            <xs:annotation>
              <xs:documentation>provides values that correspond to each enumerator</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:any namespace="##other" minOccurs="0"/>
        </xs:sequence>
        <xs:attributeGroup ref="modelID:commonAttributes"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="arrayDataType">
  <xs:sequence>
    <xs:element name="name" type="modelID:IdentifierType"/>
    <xs:element name="dataType" type="ReferenceType">
      <xs:annotation>
        <xs:documentation>identifies the datatype of an element of this array</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
    <xs:element name="enumerator" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="name" type="modelID:IdentifierType"/>
          <xs:element name="value" type="modelID:String" maxOccurs="unbounded">
            <xs:annotation>
              <xs:documentation>identifies the datatype of an element of this array</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:any namespace="##other" minOccurs="0"/>
        </xs:sequence>
        <xs:attributeGroup ref="modelID:commonAttributes"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
array</xs:documentation>

<xs:element name="encoding" type="arrayDatatypeEncodingType">
  <xs:annotation>
    <xs:documentation>describes, in detail, the encoding of the array datatype (e.g., the sequence of 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:sequence>
<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="variantRecordDataType">
  <xs:sequence>
    <xs:element name="name" type="modelID:IdentifierType"/>
    <xs:element name="discriminant" type="modelID:IdentifierType"/>
    <xs:element name="dataType" type="ReferenceType">
      <xs:annotation>
        <xs:documentation>identifies the datatype of the discriminant</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="alternative" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="enumerator" type="modelID:NonEmptyString">
            <xs:annotation>
              <xs:documentation>enumerators or enumerator ranges that determines the alternative</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element name="name" type="modelID:IdentifierType" minOccurs="0"/>
          <xs:element name="dataType" type="ReferenceType" minOccurs="0">
            <xs:annotation>
              <xs:documentation>identify the datatype of the field</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="encoding" type="variantRecordEncodingType">
      <xs:annotation>
        <xs:documentation>describes the encoding of the variant record datatype (e.g., the location of the discriminant) as delivered to and received from the RTI</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="notesType">
  <xs:sequence>
    <xs:element name="note" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="label" type="xs:ID"/>
          <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="dataTypesType">
  <xs:sequence>
    <xs:element name="name" type="modelID:IdentifierType"/>
    <xs:element name="discriminant" type="modelID:IdentifierType"/>
    <xs:element name="dataType" type="ReferenceType">
      <xs:annotation>
        <xs:documentation>identifies the datatype of the discriminant</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>
Basic data representation is the underpinning of all OMT datatypes. Although it is not used as a datatype, it forms the basis of the datatypes.

The simple datatypes describe simple, scalar data discrete set of possible values.

The array datatypes describe indexed homogeneous collections of datatypes; also known as records or structures.

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

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

Copyright © 2004-2005 SISO. All rights reserved.
This is an unapproved SISO Draft Specification, subject to change.
<xs:element name="tags" type="tagsType">
  <xs:annotation>
    <xs:documentation>specifies the datatype of user-defined tags</xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element name="synchronizations" type="synchronizationsType">
  <xs:annotation>
    <xs:documentation>specifies federate and federation capabilities for synchronization-points</xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element name="transportations" type="transportationsType">
  <xs:annotation>
    <xs:documentation>documents transportation type support and agreements</xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element name="switches" type="switchesType">
  <xs:annotation>
    <xs:documentation>specification of the initial setting of RTI switches</xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element name="updateRates"/>
<xs:element name="dataTypes" type="dataTypesType">
  <xs:annotation>
    <xs:documentation>specifies all referenced datatypes</xs:documentation>
  </xs:annotation>
</xs:element>

<xs:element name="notes" type="notesType">
  <xs:annotation>
    <xs:documentation>specifies all referenced notes</xs:documentation>
  </xs:annotation>
</xs:element>

<xs:any namespace="##other" minOccurs="0"/>
</xs:sequence>

<xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xs:complexType name="Size">
  <xs:simpleContent>
    <xs:extension base="xs:integer">
      <xs:attributeGroup ref="modelID:commonAttributes"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="RateType">
  <xs:simpleContent>
    <xs:extension base="xs:float">
      <xs:attributeGroup ref="modelID:commonAttributes"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="ReferenceType">
  <xs:simpleContent>
    <xs:extension base="xs:NCName">
      <xs:attributeGroup ref="modelID:commonAttributes"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="sharingType">
  <xs:simpleContent>
    <xs:extension base="sharingEnumerations">
      <xs:attributeGroup ref="modelID:commonAttributes"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="orderType">
  <xs:simpleContent>
    <xs:extension base="orderEnumerations">
      <xs:attributeGroup ref="modelID:commonAttributes"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="updateType">
  <xs:simpleContent>
    <xs:extension base="updateEnumerations">
      <xs:attributeGroup ref="modelID:commonAttributes"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

Copyright © 2004-2005 SISO. All rights reserved.
This is an unapproved SISO Draft Specification, subject to change.
<xs:sequence>
  <xs:element name="modelID:IdentifierType" minOccurs="0"/>
  <xs:element name="rate" type="RateType" minOccurs="0" maxOccurs="unbounded"/>
  <xs:element name="semantics" type="modelID:String" minOccurs="0"/>
  <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>

<xref:complexType>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xref:complexType name="updateRatesType">
  <xs:sequence>
    <xs:element name="updateRate" type="updateRateType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attributeGroup ref="modelID:commonAttributes"/>
</xs:complexType>

<xref:complexType name="conformanceType">
  <xs:sequence>
    <xs:element name="implementedServices">
      <xs:annotation>
        <xs:documentation>RTI services implemented/used in the federation or by a</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:documentation>Federate</xs:annotation>
</xs:complexType>

<!-- Federation Management Services -->
<xs:element name="createFederationExecution" minOccurs="0"/>
<xs:element name="destroyFederationExecution" minOccurs="0"/>
<xs:element name="joinFederationExecution" minOccurs="0"/>
<xs:element name="resignFederationExecution" minOccurs="0"/>
<xs:element name="registerFederateIdentifierRegistration" minOccurs="0"/>
<xs:element name="confirmFederateIdentifierRegistration" minOccurs="0"/>
<xs:element name="withdrawFederateIdentifierRegistration" minOccurs="0"/>
<xs:element name="registerFederateIdentifierReserved" minOccurs="0"/>
<xs:element name="confirmFederateIdentifierReserved" minOccurs="0"/>
<xs:element name="withdrawFederateIdentifierReserved" minOccurs="0"/>
<xs:element name="registerFederateIdentifier" minOccurs="0"/>
<xs:element name="confirmFederateIdentifier" minOccurs="0"/>
<xs:element name="withdrawFederateIdentifier" minOccurs="0"/>
<xs:element name="registerFederate" minOccurs="0"/>
<xs:element name="confirmFederate" minOccurs="0"/>
<xs:element name="withdrawFederate" minOccurs="0"/>
<xs:element name="registerFederateRSVP" minOccurs="0"/>
<xs:element name="confirmFederateRSVP" minOccurs="0"/>
<xs:element name="withdrawFederateRSVP" minOccurs="0"/>
<xs:element name="syncRegistration" minOccurs="0"/>
<xs:element name="syncReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
<xs:element name="timezoneReserved" minOccurs="0"/>
<xs:element name="timezoneRegistration" minOccurs="0"/>
This is an unapproved SISO Draft Specification, subject to change.
<xs:element name="getKnownObjectClassHandle" minOccurs="0"/>
<xs:element name="getAvailableDimensionsForInteractionClass" minOccurs="0"/>
<xs:element name="getTransportationType" minOccurs="0"/>
<xs:element name="getTransportationName" minOccurs="0"/>
<xs:element name="getOrderType" minOccurs="0"/>
<xs:element name="getOrderName" minOccurs="0"/>
<xs:element name="enableObjectClassRelevanceAdvisorySwitch" minOccurs="0"/>
<xs:element name="disableObjectClassRelevanceAdvisorySwitch" minOccurs="0"/>
<xs:element name="enableAttributeRelevanceAdvisorySwitch" minOccurs="0"/>
<xs:element name="disableAttributeRelevanceAdvisorySwitch" minOccurs="0"/>
<xs:element name="enableAttributeScopeAdvisorySwitch" minOccurs="0"/>
<xs:element name="disableAttributeScopeAdvisorySwitch" minOccurs="0"/>
<xs:element name="enableInteractionRelevanceAdvisorySwitch" minOccurs="0"/>
<xs:element name="disableInteractionRelevanceAdvisorySwitch" minOccurs="0"/>
<xs:element name="getDimensionHandleSet" minOccurs="0"/>
<xs:element name="getRangeBounds" minOccurs="0"/>
<xs:element name="setRangeBounds" minOccurs="0"/>
<xs:element name="normalizeFederateHandle" minOccurs="0"/>
<xs:element name="normalizeServiceGroup" minOccurs="0"/>
<xs:element name="evokeCallback" minOccurs="0"/>
<xs:element name="evokeMultipleCallback" minOccurs="0"/>
<xs:element name="enableCallbacks" minOccurs="0"/>
<xs:element name="disableCallbacks" minOccurs="0"/>
<xs:element name="getAutomaticResignDirective" minOccurs="0"/>
<xs:element name="setAutomaticResignDirective" minOccurs="0"/>
<xs:element name="getFederateHandle" minOccurs="0"/>
<xs:element name="getFederateName" minOccurs="0"/>
<xs:element name="getUpdateRateValueForAttribute" minOccurs="0"/>

</xs:complexType>
</xs:element>
</xs:complexType>
<xs:simpleType name="dimensionValuePattern">
  <xs:restriction base="xs:string">
    <xs:pattern value="(\d)+|[(\d)+..(\d)+]|Excluded"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="cardinalityPattern">
  <xs:restriction base="xs:string">
    <xs:pattern value="(Dynamic|\[(\d)+..(\d)+\]|\[(\d)+..(\d)+\]|\[Excluded\])"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="cardinalityType">
  <xs:simpleContent>
    <xs:extension base="cardinalityPattern">
      <xs:attributeGroup ref="modelID:commonAttributes"/>
    </xs:extension>
  </xs:simpleContent>
</xs:simpleType>
<xs:simpleType name="arrayDatatypeEncodingEnum">
  <xs:restriction base="xs:string">
    <xs:pattern value="HLAfixedArray"/>
    <xs:pattern value="HLAvariableArray"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="arrayDatatypeEncodingUnion">
  <xs:union memberTypes="arrayDatatypeEncodingEnum modelID:nonEmptyString"/>
</xs:simpleType>
<xs:simpleType name="arrayDatatypeEncodingType">
  <xs:simpleContent>
    <xs:extension base="arrayDatatypeEncodingUnion">
      <xs:attributeGroup ref="modelID:commonAttributes"/>
    </xs:extension>
  </xs:simpleContent>
</xs:simpleType>
<xs:simpleType name="fixedRecordEncodingEnumeration">
  <xs:restriction base="xs:string">
    <xs:enumeration value="HLAfixedRecord"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="fixedRecordEncodingUnion">
  <xs:union memberTypes="fixedRecordEncodingEnumeration modelID:nonEmptyString"/>
</xs:simpleType>
<xs:simpleType name="fixedRecordEncodingType">
  <xs:simpleContent>
    <xs:extension base="fixedRecordEncodingUnion">
      <xs:attributeGroup ref="modelID:commonAttributes"/>
    </xs:extension>
  </xs:simpleContent>
</xs:simpleType>

Copyright © 2004-2005 SISO. All rights reserved.
This is an unapproved SISO Draft Specification, subject to change.
<xs:extension base="fixedRecordEncodingUnion">
  <xs:attributeGroup ref="modellD:commonAttributes"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:simpleType>
<xs:complexType name="variantRecordEncodingType">
  <xs:simpleContent>
    <xs:extension base="variantRecordEncodingUnion">
      <xs:attributeGroup ref="modellD:commonAttributes"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
</xs:schema>
9 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 v5 rel. 4 U (http://www.xmlspy.com) by Bjorn Lofstrand (Pitch Technologies) -->
    xsi:schemaLocation="http://www.sisostds.org/schemas/bom 0_12.xsd BOM_v">
    <modelIdentification>
        <modelID:modificationDate>2003-08-27</modelID:modificationDate>
        <modelID:name>RestaurantPayment34</modelID:name>
        <modelID:type>BOM</modelID:type>
        <modelID:version>1.0 Beta3</modelID:version>
        <modelID:securityClassification>Unclassified</modelID:securityClassification>
        <modelID:releaseRestriction>Not for release outside the XYZ Restaurant Corporation.</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>business type</modelID:taxonomy>
            <modelID:keywordValue>restaurant</modelID:keywordValue>
        </modelID:keyword>
        <modelID:poc>
            <modelID:pocType>Primary author</modelID:pocType>
            <modelID:pocName>Mr. Snuffy Smith</modelID:pocName>
            <modelID:pocOrg>XYZ Restaurant Corporation</modelID:pocOrg>
            <modelID:pocTelephone>+1 44 123-456-7890</modelID:pocTelephone>
            <modelID:pocEmail>snuffy.smith@goodeats.here.com</modelID:pocEmail>
        </modelID:poc>
        <modelID:poc>
            <modelID:pocType>Release authority</modelID:pocType>
            <modelID:pocName>XYZ Restaurant Corporation</modelID:pocName>
            <modelID:pocOrg>XYZ Restaurant Corporation</modelID:pocOrg>
            <modelID:pocTelephone>+1 44 123-456-1000</modelID:pocTelephone>
            <modelID:pocEmail>the.lawyer@xyz-foods.com</modelID:pocEmail>
        </modelID:poc>
        <modelID:other>Copyright © 2004-2005 SISO. All rights reserved. This is an unapproved SISO Draft Specification, subject to change.</modelID:other>
    </modelIdentification>
    <conceptualModel>
        <patternDescription>
            <name>PaymentPattern</name>
            <action>
                <sequence>1</sequence>
                <event>WaiterNotified</event>
                <sender>Customer</sender>
                <receiver>Waiter</receiver>
                <semantics>Waiter is made aware that the customer will not order anything else</semantics>
            </variation>
            <name>WaiterObservesDirtyDishes</name>
            <sender>Table</sender>
            <receiver>Waiter</receiver>
            <semantics>Waiter observes that the table is full of dirty dishes</semantics>
        </conceptualModel>
    </conceptualModel>
</BOM>
```
<eventType>
  <name>CheckRequestMessage</name>
  <sourceCharacteristic>
    <name>CustomerID</name>
  </sourceCharacteristic>
  <targetCharacteristic>
    <name>WaiterID</name>
  </targetCharacteristic>
  <contentCharacteristic>
    <name>CheckRequest</name>
  </contentCharacteristic>
  <semantics>Request check from waiter</semantics>
</eventType>

<eventType>
  <name>PaymentDueNotificationMessage</name>
  <sourceCharacteristic>
    <name>WaiterID</name>
  </sourceCharacteristic>
  <targetCharacteristic>
    <name>CustomerID</name>
  </targetCharacteristic>
  <contentCharacteristic>
    <name>Amount</name>
  </contentCharacteristic>
  <semantics>Notification of Payment Due</semantics>
</eventType>

<eventType>
  <name>PaymentByCashMessage</name>
  <sourceCharacteristic>
    <name>CustomerID</name>
  </sourceCharacteristic>
  <targetCharacteristic>
    <name>WaiterID</name>
  </targetCharacteristic>
  <contentCharacteristic>
    <name>Amount</name>
  </contentCharacteristic>
  <semantics>Give cash to waiter</semantics>
</eventType>

<eventType>
  <name>PaymentByCreditCardMessage</name>
  <sourceCharacteristic>
    <name>CustomerID</name>
  </sourceCharacteristic>
  <targetCharacteristic>
    <name>WaiterID</name>
  </targetCharacteristic>
  <contentCharacteristic>
    <name>CreditCardNumber</name>
  </contentCharacteristic>
  <semantics>Present Credit Card to waiter</semantics>
</eventType>

<eventType>
  <name>DirtyDishesOnTableTrigger</name>
  <sourceCharacteristic>
    <name>TableID</name>
  </sourceCharacteristic>
  <triggerCondition>TableHasDirtyDishes</triggerCondition>
  <semantics>Trigger when there is dirty dishes on the table</semantics>
</eventType>

<eventType>
  <name>NonPayingCustomerTrigger</name>
  <sourceCharacteristic>
    <name>CustomerID</name>
  </sourceCharacteristic>
  <triggerCondition>CustomerLeavesAndHasNotPaid</triggerCondition>
  <semantics>Trigger when customer leaves without paying</semantics>
</eventType>

<conceptualModel>
  <modelMapping>
    <eventTypeMappings>
      <mappedType>
        <name>CheckRequestMessage</name>
        <hlaClass>HLAInteractionRoot.RealWorldMessage.RequestCheck</hlaClass>
        <characteristic>
          <name>WaiterID</name>
          <hlaProperty>HLAInteractionRoot.RealWorldMessage.RequestCheck.waiterId</hlaProperty>
        </characteristic>
      </mappedType>
    </eventTypeMappings>
  </modelMapping>
</conceptualModel>
<characteristic>
  <name>CustomerID</name>
  <hlaProperty>HLAInteractionRoot.RealWorldMessage.RequestCheck.customerId</hlaProperty>
</characteristic>

<characteristic>
  <name>CheckRequest</name>
  <hlaProperty>HLAInteractionRoot.RealWorldMessage.RequestCheck</hlaProperty>
</characteristic>

<characteristic>
  <name>PaymentDueNotificationMessage</name>
  <hlaClass>HLAInteractionRoot.RealWorldMessage.PaymentDue</hlaClass>
  <characteristic>
    <name>WaiterId</name>
    <hlaProperty>HLAInteractionRoot.RealWorldMessage.PaymentDue.waiterId</hlaProperty>
  </characteristic>
  <characteristic>
    <name>CustomerId</name>
    <hlaProperty>HLAInteractionRoot.RealWorldMessage.PaymentDue.customerId</hlaProperty>
  </characteristic>
  <characteristic>
    <name>Amount</name>
    <hlaProperty>HLAInteractionRoot.RealWorldMessage.PaymentDue.amount</hlaProperty>
  </characteristic>
</mappedType>

<characteristic>
  <name>PaymentByCashMessage</name>
  <hlaClass>HLAObjectRoot.AbstractEntity.CashPayment</hlaClass>
  <characteristic>
    <name>WaiterId</name>
    <hlaProperty>HLAObjectRoot.AbstractEntity.CashPayment.waiterId</hlaProperty>
  </characteristic>
  <characteristic>
    <name>CustomerId</name>
    <hlaProperty>HLAObjectRoot.AbstractEntity.CashPayment.customerId</hlaProperty>
  </characteristic>
  <characteristic>
    <name>Amount</name>
    <hlaProperty>HLAObjectRoot.AbstractEntity.CashPayment.amount</hlaProperty>
  </characteristic>
</mappedType>

<characteristic>
  <name>PaymentByCreditCardMessage</name>
  <hlaClass>HLAInteractionRoot.PhysicalEvent.PresentCreditCard</hlaClass>
  <characteristic>
    <name>WaiterId</name>
    <hlaProperty>HLAInteractionRoot.PhysicalEvent.PresentCreditCard.waiterId</hlaProperty>
  </characteristic>
  <characteristic>
    <name>CustomerId</name>
    <hlaProperty>HLAInteractionRoot.PhysicalEvent.PresentCreditCard.customerId</hlaProperty>
  </characteristic>
  <characteristic>
    <name>CreditCardNumber</name>
    <hlaProperty>HLAInteractionRoot.PhysicalEvent.PresentCreditCard.cardNumber</hlaProperty>
  </characteristic>
</mappedType>

<characteristic>
  <name>DirtyDishesOnTableTrigger</name>
  <hlaClass>HLAObjectRoot.PhysicalEntity.Table</hlaClass>
  <characteristic>
    <name>TableId</name>
    <hlaProperty>HLAObjectRoot.PhysicalEntity.Table.id</hlaProperty>
  </characteristic>
</modelMapping>
<objectModelDefinition>
  <objects>
    <omt:objectClass>
      <omt:name>HLAObjectRoot</omt:name>
      <omt:sharing>Neither</omt:sharing>
      <omt:attribute>
        <omt:name>HLAprivilegeToDeleteObject</omt:name>
        <omt:dataType>HLAboolean</omt:dataType>
        <omt:updateType>Static</omt:updateType>
        <omt:ownership>NoTransfer</omt:ownership>
        <omt:sharing>Subscriber</omt:sharing>
        <omt:transportation>HLAReliable</omt:transportation>
      </omt:attribute>
    </omt:objectClass>
  </objects>
</objectModelDefinition>
<omt:objectClass>
  <omt:name>PhysicalEntity</omt:name>
  <omt:sharing>PublishSubscribe</omt:sharing>
  <omt:objectClass>
    <omt:name>Table</omt:name>
    <omt:sharing>PublishSubscribe</omt:sharing>
    <omt:attribute>
      <omt:name>hasDirtyDishes</omt:name>
      <omt:dataType>Boolean</omt:dataType>
      <omt:updateType>Conditional</omt:updateType>
      <omt:updateCondition>onChange or request</omt:updateCondition>
      <omt:ownership>DivestAcquire</omt:ownership>
      <omt:sharing>PublishSubscribe</omt:sharing>
      <omt:transportation>HLA</omt:transportation>
      <omt:order>Receive</omt:order>
    </omt:attribute>
    <omt:attribute>
      <omt:name>id</omt:name>
      <omt:dataType>ID</omt:dataType>
      <omt:updateType>Static</omt:updateType>
      <omt:ownership>DivestAcquire</omt:ownership>
      <omt:sharing>PublishSubscribe</omt:sharing>
      <omt:transportation>HLA</omt:transportation>
      <omt:order>Receive</omt:order>
    </omt:attribute>
  </omt:objectClass>
  <omt:attribute>
    <omt:name>hasPaid</omt:name>
    <omt:dataType>Boolean</omt:dataType>
    <omt:updateType>Conditional</omt:updateType>
    <omt:updateCondition>onChange or request</omt:updateCondition>
    <omt:ownership>DivestAcquire</omt:ownership>
    <omt:sharing>PublishSubscribe</omt:sharing>
    <omt:transportation>HLA</omt:transportation>
    <omt:order>Receive</omt:order>
  </omt:attribute>
</omt:objectClass>

<omt:objectClass>
  <omt:name>Customer</omt:name>
  <omt:sharing>PublishSubscribe</omt:sharing>
  <omt:attribute>
    <omt:name>hasPaid</omt:name>
    <omt:dataType>Boolean</omt:dataType>
    <omt:updateType>Conditional</omt:updateType>
    <omt:updateCondition>onChange or request</omt:updateCondition>
    <omt:ownership>DivestAcquire</omt:ownership>
    <omt:sharing>PublishSubscribe</omt:sharing>
    <omt:transportation>HLA</omt:transportation>
    <omt:order>Receive</omt:order>
  </omt:attribute>
  <omt:attribute>
    <omt:name>id</omt:name>
    <omt:dataType>ID</omt:dataType>
    <omt:updateType>Static</omt:updateType>
    <omt:ownership>DivestAcquire</omt:ownership>
    <omt:sharing>PublishSubscribe</omt:sharing>
    <omt:transportation>HLA</omt:transportation>
    <omt:order>Receive</omt:order>
  </omt:attribute>
</omt:objectClass>

<omt:objectClass>
  <omt:name>AbstractEntity</omt:name>
  <omt:sharing>PublishSubscribe</omt:sharing>
  <omt:objectClass>
    <omt:name>CashPayment</omt:name>
    <omt:sharing>PublishSubscribe</omt:sharing>
    <omt:attribute>
      <omt:semantics>Represents a payment using cash.</omt:semantics>
      <omt:attribute>
        <omt:name>amount</omt:name>
        <omt:dataType>Float32BE</omt:dataType>
        <omt:updateType>Conditional</omt:updateType>
        <omt:updateCondition>onChange or request</omt:updateCondition>
        <omt:ownership>DivestAcquire</omt:ownership>
        <omt:sharing>PublishSubscribe</omt:sharing>
        <omt:transportation>HLA</omt:transportation>
        <omt:order>Receive</omt:order>
      </omt:attribute>
      <omt:attribute>
        <omt:name>customerID</omt:name>
        <omt:dataType>ID</omt:dataType>
        <omt:updateType>Static</omt:updateType>
        <omt:ownership>DivestAcquire</omt:ownership>
        <omt:sharing>PublishSubscribe</omt:sharing>
        <omt:transportation>HLA</omt:transportation>
        <omt:order>Receive</omt:order>
      </omt:attribute>
      <omt:attribute>
        <omt:name>waiterId</omt:name>
        <omt:dataType>ID</omt:dataType>
        <omt:updateType>Static</omt:updateType>
        <omt:ownership>DivestAcquire</omt:ownership>
        <omt:sharing>PublishSubscribe</omt:sharing>
        <omt:transportation>HLA</omt:transportation>
        <omt:order>Receive</omt:order>
      </omt:attribute>
    </omt:objectClass>
  </omt:attribute>
</omt:objectClass>

Copyright © 2004-2005 SISO. All rights reserved.
This is an unapproved SISO Draft Specification, subject to change.
## 10 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 XMLSpy tool. This section describes the symbols and nomenclature found within these diagrams beginning with the Element Symbols identified in Table C-1.\(^7\)

### Table C-1 – Element Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Country" /></td>
<td>Mandatory single element</td>
<td>The rectangle indicates an element and the solid border indicates that the element is required. MinOcc=1, MaxOcc=1</td>
</tr>
<tr>
<td><img src="image" alt="Name" /></td>
<td>Mandatory single element, containing Parsed Character Data (#PC-Data).</td>
<td>The content may be simple content or mixed complex content. MinOcc=1, MaxOcc=1, type=xsd:string, content=simple.</td>
</tr>
<tr>
<td><img src="image" alt="Location" /></td>
<td>Single optional element</td>
<td>The context menu option Optional, converts a mandatory element into an optional one. MinOcc=0, MaxOcc=1</td>
</tr>
<tr>
<td><img src="image" alt="Alias" /></td>
<td>Mandatory multiple element</td>
<td>MinOcc=1, MaxOcc=5</td>
</tr>
<tr>
<td><img src="image" alt="Division" /></td>
<td>Mandatory multiple element containing child elements.</td>
<td>MinOcc=1, MaxOcc=unbounded, type=DivisionType, content=complex.</td>
</tr>
<tr>
<td><img src="image" alt="xs:field" /></td>
<td>Element referencing global element</td>
<td>Arrow in bottom-left indicates element referencing global element. MinOcc=1, MaxOcc=unbounded</td>
</tr>
<tr>
<td><img src="image" alt="keybase" /></td>
<td>Complex Type</td>
<td>Represented using an irregular hexagon</td>
</tr>
<tr>
<td><img src="image" alt="Subsidiaries" /></td>
<td>Model Group</td>
<td>Represented using an irregular octagon with a plus sign. A model group allows you to define and reuse element declarations.</td>
</tr>
<tr>
<td><img src="image" alt="any ##other" /></td>
<td>Wildcards</td>
<td>The irregular octagon with any at left indicates a wildcard. Can be a placeholder for any element from a certain namespace.</td>
</tr>
</tbody>
</table>


Copyright © 2004-2005 SISO. All rights reserved. This is an unapproved SISO Draft Specification, subject to change.
Attributes

Indicated with the word ‘attributes’ in italics in a rectangle that can be expanded. Each attribute is shown in a rectangle with a dashed border.

Identity constraints

Indicated with the word ‘constraints’ in italics in a rectangle that can be expanded.

Simple types

A "simple type" element is defined as a data type that only contains values and no element or attributes. The element type is usually prefixed by the namespace prefix `xsd: string`, indicating that it is a predefined XML Schema data type.

Details: name=Name, type=xsd:string, content=simple.

Complex types

"Complex type" is a data type which may contain attributes, elements and text. Adding sub elements to an element, automatically defines the element with the content model as complex.

Figure C-1 – Complex Types

The keybase complex type shown above 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 element reference global elements of those names.)

Compositors

Copyright © 2004-2005 SISO. All rights reserved.
This is an unapproved SISO Draft Specification, subject to change.
A "Compositor" defines an ordered sequence of sub elements (child elements). Compositors are identified in Table C-2.

### Table C-2 - Compositors

<table>
<thead>
<tr>
<th>Compositor</th>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sequence</strong></td>
<td><img src="image" alt="Sequence Diagram" /></td>
</tr>
<tr>
<td><strong>Choice</strong></td>
<td><img src="image" alt="Choice Diagram" /></td>
</tr>
<tr>
<td><strong>All</strong></td>
<td><img src="image" alt="All Diagram" /></td>
</tr>
</tbody>
</table>