3.0 Introduction

The HLA requires that federations and individual federates be described by an object model which identifies the data exchanged at runtime in order to achieve federation objectives. The documentation of the object model is defined by the HLA Object Model Template (OMT) Specification, which serves as one of the three related standards for the HLA. The HLA Rules 1 and 6, which are reproduced in Figure 3.1, refer specifically to the HLA Object Model Template.

Review of HLA Rules

• Rule 1: Federations shall have an HLA federation object model (FOM), documented in accordance with the HLA object model template (OMT).

• Rule 6: Federates shall have an HLA simulation object model (SOM), documented in accordance with the HLA object model template.

The HLA OMT Specification prescribes the format and syntax for recording information in HLA object models, to include objects, attributes, interactions and parameters, but it does not define the specific data that will appear in the object model. Thus the HLA offers the flexibility of operating across different computing environments using a variety of programming languages.
The HLA OMT Specification provides (among other things) the rationale for using a standardized structural framework, or template, for specifying HLA object models (see Figure 3.2).

**Rationale for an Object Model Template**

- Provides a commonly understood mechanism for specifying the exchange of data and general coordination among the members of a federation.
- Provides a common, standardized mechanism for describing the capabilities of potential federation members.
- Facilitates the design and application of common tool sets for development of HLA object models.

**Figure 3.2 Rationale for an Object Model Template**

HLA object models may be used to describe an individual federation member (federate); to create an HLA simulation object model (SOM); or to describe a named set of multiple interacting federates (federation) by creating a federation object model (FOM).

The primary function of an HLA FOM is to specify, in a common standardized format, the nature of the data exchange among federates. This data includes an enumeration of all object and interaction classes pertinent to the federation, along with a specification of the attributes or parameters characterizing these classes.

An HLA SOM is a specification of the types of information that an individual federate could provide to HLA federations and the information that an individual federate could receive from other federates in HLA federations. The standard format in which SOMs are expressed facilitates determination of the suitability of federates for participation in a federation.
Note that there are differences between the way object models are defined in the HLA and in the techniques of Object-Oriented Analysis and Design (OOAD). In OOAD an object model provides a much wider description of an object including complete details of the internal relationships characterizing the object. HLA object models, on the other hand, are much narrower, focusing on federate/federation information interchange.

The following information about the composition of the OMT applies equally to FOMs and SOMs unless specified otherwise.

3.1 Components of the OMT

HLA object models are composed of a group of interrelated components specifying information about classes of objects and their attributes, and interactions and their parameters. The information content of these components can be represented in many different ways or presentations. A presentation is the formatting of the information contained in the object model in a particular manner for a particular purpose. For example, the OMT tabular format is designed for presentation on a printed page, while the OMT Data Interchange Format (DIF) is a presentation designed for passing an object model between tools. All HLA object models shall be capable of being presented in both the OMT tabular format and the OMT DIF format. We will discuss the OMT content and present it in the OMT tabular format.

The OMT consists of the following fourteen components:

— Object model identification table
— Object class structure table
— Interaction class structure table
— Attribute table
— Parameter table
— Dimension table
— Time representation table
— User-supplied tag table
— Synchronization table
— Transportation type table
— Switches table
— Datatype tables
— Notes table
— FOM/SOM lexicon
The OMT Components

• Object model identification table
• Object class structure table
• Interaction class structure table
• Attribute table
• Parameter table
• Dimension table
• Time representation table

The OMT Components (cont’d)

• User-supplied tag table
• Synchronization table
• Transportation type table
• Switches table
• Datatype tables
• Notes table
• FOM/SOM lexicon

According to the OMT Specification, all of the OMT components shall be completed when specifying an HLA object model for both federations and individual federates. However, certain tables may be empty or devoid of domain-specific content. While
federations typically support interactions among their federates, some federates might not be involved in interactions. In this situation, the interaction class structure table would contain only the single interaction class required by the HLA and the parameter table would be empty in that federate’s SOM. It is also expected that federates commonly have objects with attributes of interest across the federation; in such cases, these objects and attributes shall be documented. However, a federate or an entire federation may exchange information solely via interactions, in which case its object class structure table and attribute table would contain only HLA-required data.

The components of the OMT are now described briefly. More complete descriptions can be found in the HLA OMT Specification.

### 3.1.1 Object Model Identification Table

A key design goal for all HLA object models is to facilitate reuse. HLA object models provide information that enables inferences to be drawn regarding the reuse potential of individual federates for new applications. Reuse can also occur at the level of the object model itself. An existing FOM may provide a foundation for the development of new FOMs, or applicable components of existing SOMs may be merged to form new FOMs. In either case, to expedite reuse, it is important to include a minimum but sufficient degree of descriptive information in the object model. When federation developers pose detailed questions about how a federate or federation was constructed, point-of-contact (POC) information within an HLA object model is very important. The purpose of the object model identification table is to document certain key identifying information within the object model description itself. Figure 3.4 illustrates the structure of the object model identification table and provides an example of its contents, which are largely self-explanatory.

Thus the object model identification table has two columns. The first column (Category) specifies the categories of data that shall be provided in this table. The second column (Information) shall specify the required information. Entries shall be provided for all rows except References and other; “NA” shall be entered for these rows if no information is appropriate.
### Sample Object Model Identification Table

<table>
<thead>
<tr>
<th>Category</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Restaurant: Example</td>
</tr>
<tr>
<td>Type</td>
<td>SOM</td>
</tr>
<tr>
<td>Version</td>
<td>1.0 Alpha</td>
</tr>
<tr>
<td>Modification Date</td>
<td>1-Jan-98</td>
</tr>
<tr>
<td>Purpose</td>
<td>Example object model for restaurant federate</td>
</tr>
<tr>
<td>Application Domain</td>
<td>Restaurant operations</td>
</tr>
<tr>
<td>Sponsor</td>
<td>Federated foods</td>
</tr>
<tr>
<td>POC</td>
<td>Mr. Joseph Doe</td>
</tr>
<tr>
<td>POC Organisation</td>
<td>Joe's Place</td>
</tr>
<tr>
<td>POC Telephone</td>
<td>(977)555 1234</td>
</tr>
<tr>
<td>POC Email</td>
<td><a href="mailto:doejj@fedfoods.com">doejj@fedfoods.com</a></td>
</tr>
<tr>
<td>References</td>
<td><a href="http://www.fedfoods.com/restsim.html">www.fedfoods.com/restsim.html</a></td>
</tr>
<tr>
<td>Other</td>
<td>See Mobil Intl. Restaurant Guide</td>
</tr>
</tbody>
</table>

---

#### 3.1.2 Object Class Structure Table

An HLA object class is a collection of objects with certain characteristics or attributes in common. An HLA object class structure shall be defined in terms of hierarchical relationships among classes of objects. Class relationships shall be represented by the inclusion of the associated class names in the appropriate columns of the object class structure table. Relationships among classes at non-adjacent levels of a class hierarchy can be derived through transitivity: if A is a superclass of B and B is a superclass of C, then A is also a superclass of C. Note that A is a superclass of B if and only if B is a subclass of A.

Subclasses can be considered to be specializations, or refinements, of their superclasses. Subclasses shall always inherit the attributes of their superclasses, and may possess additional attributes to provide the desired specialization.

The object class "HLAObjectRoot" shall be a superclass of all other object classes in a FOM or SOM. The HLA object model shall support only single inheritance; in this mechanism, each class has at most one immediate superclass.

The object class structure table shall be filled by first entering the root class of all object classes in the left-most column; this object class is named "HLAObject Root." Then the most general object classes shall be entered in the next column, followed by all their
immediate subclasses in the next column, etc. Except for the required root class, the HLA does not require any other specific object classes to appear in the object class structure table.

Figure 3.5 shows a partially completed example of the contents of the object class structure table.

![Sample Object Class Structure Table](image)

**Figure 3.5 Object Class Structure Table**

Each object class is classified according to its publication and subscription capabilities. The letters P, S and N indicate publishable, subscribable and neither publishable nor subscribable respectively. A class can be classified as both publishable and subscribable using the letters PS.

Publishable means that a federate using the Publish Object Class Attributes service can publish the specified object class. Subscribable means that a federate is capable of utilizing and reacting to information on objects in the given class.
3.1.3 Interaction Class Structure Table

An interaction is defined as an explicit action taken by a federate that may have some effect or impact on another federate within a federation execution. Interactions shall be specified in the interaction class structure table of HLA object models in terms of their class-subclass relationships, in much the same way that objects are described in the object class structure table.

An interaction hierarchy in an HLA FOM is primarily designed to support inheritance. In SOMs, an interaction hierarchy reflects how the federate supports publication and subscription of interaction classes. The interaction class $\text{HLAinteractionRoot}$ shall be a superclass of all other interaction classes in a FOM or SOM.

The basic format for the interaction class structure table follows the format for the object class structure table. In particular, the root interaction class shall be specified in the left-most column and shall be named $\text{HLAinteractionRoot}$; this is the only required class. Subsequent columns to the right shall contain (if necessary) interaction classes with increasing degrees of class specificity.

Figure 3.6 shows a partially completed example of the interaction class structure table.

![Sample Interaction Class Structure Table](image)

**Figure 3.6 Interaction Class Structure Table**

Each interaction class in the interaction class structure table shall be followed by information on publishing and subscribing capabilities enclosed in parentheses. Let us
temporarily denote that type of information by a variable \(<p/s>\).

For a SOM, valid entries for \(<p/s>\) shall be as follows:
- \(P\) (Publish): The federate is capable of publishing the interaction class.
- \(S\) (Subscribe): The federate is capable of subscribing to the interaction class.
- \(PS\) (Publish/Subscribe): The federate is capable of publishing and subscribing to the interaction class.
- \(N\) (Neither): The federate is incapable of either publishing or subscribing to the interaction class.

For a FOM, the same entries for \(<p/s>\) are valid. An interaction class is classified as \(Publish\) or \(Subscribe\) in the federation if at least one federate is capable of publishing or subscribing to the class.

Classes designated as \(Subscribe\) or \(Neither\) are never sent but they can have subclasses that are sent.

### 3.1.4 Attribute Table

Each class of simulation domain objects is characterized by a fixed set of attribute types. These attributes are named portions of the state of their object whose values can change over time (such as the velocity of a vehicle). The values of HLA instance attributes are updated through the RTI and provided to other federates in a federation. Both federates and federations shall document all such object attributes in the attribute table of their SOM or FOM. The object class \(HLAobjectRoot\), which shall be a superclass of all other object classes, may have attributes assigned to it like to any other object class.

The attribute table of a FOM describes all object attributes represented in a federation. The attribute table of a SOM describes all class attributes that are published or subscribed to by the federate.

The first column (Object) contains names from the object class structure table for object classes that are associated with attributes.

The second column (Attribute) lists the attributes of the specified object class.

The third column (Datatype) identifies the datatype of the attribute. Valid datatypes are described in detail in Section 4.12 (Datatype tables) of the IEEE P1516.2 Standard. For each datatype used in the federate, various details (such as resolution and accuracy) must be provided in separate datatype tables. An attribute’s datatype can be specified “NA” but that attribute must still have valid (non-“NA”) transportation and order types specified. When a datatype of “NA” is used for an attribute, its update type, update
condition and available dimensions shall also be “NA.”

The names of the remaining seven columns are: Update Type, Update Condition, D/A, P/S, Available Dimensions, Transportation, Order. (We already referred to some of them in the preceding paragraph.)

The Update Type column records the policy for updating an instance of the class attribute. The unique designations for this column shall be as follows:

- **Static**: The value of the attribute is static; the federate updates it initially and when requested.
- **Periodic**: The federate updates the attribute at regular time intervals.
- **Conditional**: The federate updates the attribute when unique conditions dictate.
- **NA**: The federate never provides a value for this attribute.

The Update Condition column shall expand and explain the policies for updating an instance of the class attribute. For example, if the update type is periodic, a rate of number of updates per time-unit shall be specified in this column and if the update type is conditional, this column shall specify conditions for update. If the update type is Static or N/A, “N/A” shall be entered in this column.

The D/A column shall indicate whether ownership of an instance of the class attribute can be divested or acquired. The unique designations for this column are as follows:

- **D** (Divest)
- **A** (Acquire)
- **N** (No-Transfer)
- **DA** (DivestAcquire)

The P/S column shall identify the capabilities of a federate or federation with respect to class attribute publishing and subscribing.

The Available Dimensions column shall record the association of the class attribute with a set of dimensions if a federate or federation is using Data Distribution Management (DDM) services for this attribute.

The Transportation column shall specify how the data (interactions and object instance values) is to be transported among federates. There are two transportation types provided by all RTI implementations: HLAreliable and HLAbestEffort. Other types may be provided by specific RTIs.

The order column shall specify the order of delivery to be used with instances of this class attribute. Valid entries are:

- **Receive**: Instances of the class attribute are delivered to a receiving federate in
an undetermined order.
— TimeStamp: Instances of the class attribute are delivered to a receiving federate in an order determined by assigned time stamps.

Figure 3.7 shows a partially completed example of the attribute table.

### Sample Attribute Table

<table>
<thead>
<tr>
<th>Object</th>
<th>Attribute</th>
<th>Data-type</th>
<th>Update Type</th>
<th>Update Condition</th>
<th>P/S</th>
<th>Available Dimensions</th>
<th>Transporta-</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLAObject Root</td>
<td>HLA privilege</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>N</td>
<td>NA</td>
<td>HLAreliable</td>
<td>Time Stamp</td>
</tr>
<tr>
<td></td>
<td>toDelete Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PayRate</td>
<td>Dollar Rate</td>
<td>Conditional</td>
<td>Merit Increase *[^1,2]</td>
<td>DA</td>
<td>PS</td>
<td>NA</td>
<td>HLAreliable</td>
<td>Time Stamp</td>
</tr>
<tr>
<td>YearsOf Service</td>
<td>Years</td>
<td>Periodic</td>
<td>1/year *[^2]</td>
<td>DA</td>
<td>PS</td>
<td>NA</td>
<td>HLAreliable</td>
<td>Time Stamp</td>
</tr>
<tr>
<td>Home Number</td>
<td>HLAASCII string</td>
<td>Conditional</td>
<td>Employee request</td>
<td>DA</td>
<td>PS</td>
<td>NA</td>
<td>HLAreliable</td>
<td>Time Stamp</td>
</tr>
<tr>
<td>Home Address</td>
<td>Address Type</td>
<td>Conditional</td>
<td>Employee request</td>
<td>DA</td>
<td>PS</td>
<td>NA</td>
<td>HLAreliable</td>
<td>Time Stamp</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Waiter Value</td>
<td>Conditional</td>
<td>Performance review</td>
<td>DA</td>
<td>PS</td>
<td>NA</td>
<td>HLAreliable</td>
<td>Time Stamp</td>
</tr>
<tr>
<td>Cheerfulness</td>
<td>Waiter Value</td>
<td>Conditional</td>
<td>Performance review</td>
<td>DA</td>
<td>PS</td>
<td>NA</td>
<td>HLAreliable</td>
<td>Time Stamp</td>
</tr>
<tr>
<td>State</td>
<td>Waiter Tasks</td>
<td>Conditional</td>
<td>Workflow</td>
<td>DA</td>
<td>PS</td>
<td>NA</td>
<td>HLAreliable</td>
<td>Time Stamp</td>
</tr>
<tr>
<td>Food.Drink</td>
<td>Number Cups</td>
<td>Conditional</td>
<td>Customer request</td>
<td>N</td>
<td>PS</td>
<td>BarQuantity</td>
<td>HLAreliable</td>
<td>Time Stamp</td>
</tr>
<tr>
<td></td>
<td>Drink Count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Food.Drink Soda</td>
<td>Flavor Type</td>
<td>Conditional</td>
<td>Customer request</td>
<td>N</td>
<td>PS</td>
<td>SodaFlavor, BarQuantity</td>
<td>HLAreliable</td>
<td>Time Stamp</td>
</tr>
</tbody>
</table>

Figure 3.7 Attribute Table

### 3.1.5 Parameter Table

For every interaction class identified in the interaction class structure table, the full set of parameters associated with that interaction class is specified in the parameter table. Recall that the interaction class HLAinteractionRoot is a superclass of all other interaction classes in a FOM or SOM; parameters may be associated with an interaction class at any level of an interaction class hierarchy. An interaction class inherits the parameters defined for its superclasses.

The parameter table shall consist of the following six columns:

1. Interaction: contains names of interaction classes from the interaction class structure table.
2. Parameter: lists the parameters of each interaction.
3. Datatype: identifies the data type of the parameter.
4. Available Dimensions: records the association of an interaction class with a set of
dimensions if the federate or federation is using DDM services for this interaction.
5. **Transportation**: specifies the type of transportation to be used with this interaction.
6. **Order**: specifies the order of delivery to be used with this interaction. Valid values for entry in this column are `Receive` and `TimeStamp`.

Figure 3.8 shows a partially completed example of the parameter table.

### Sample Parameter Table

<table>
<thead>
<tr>
<th>Interaction</th>
<th>Parameter</th>
<th>Datatype</th>
<th>Available Dimensions</th>
<th>Transportation</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomerSeated</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>HLAreliable</td>
<td>TimeStamp</td>
</tr>
<tr>
<td>FoodServed.MainCourseServed</td>
<td>TemperatureOK</td>
<td>ServiceStat</td>
<td>HLAboolean</td>
<td>WaiterId</td>
<td>HLAreliable</td>
</tr>
<tr>
<td>FoodServed.MainCourseServed</td>
<td>AccuracyOK</td>
<td>ServiceStat</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FoodServed.MainCourseServed</td>
<td>TimelinessOK</td>
<td>HLAboolean</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.8 Parameter Table**

### 3.1.6 Dimension Table

Federations use declaration management (DM) services to enable the flow of instance attribute and interaction data between federates and to limit the delivery of some data on the basis of object class, interaction class, and attribute name. This reduction of data may be insufficient to meet the needs of federations with large numbers of federates, large numbers of objects or interactions in classes, or large numbers of instance attribute updates per object. Such federations can use Data Distribution Management (DDM) services to further reduce the volume of data delivered to federates. In these cases, a common framework is required for specifying the data distribution model for the federation. Dimensions are the most fundamental DDM concept. Each attribute or interaction with which DDM services can be used will have a set of available dimensions, as indicated in the attribute table or parameter table,
respectively. Each set of available dimensions is a subset of all dimensions that are defined in the dimension table, and defines a multidimensional coordinate system through which federates either express an interest in receiving data or declare their intention to send data. These intentions are specified by creating regions within the coordinate system that indicate particular areas of interest. These regions are then used in one of two ways:

1. Subscription regions: sets of ranges that narrow the scope of interest of the subscribing federate.
2. Update regions: sets of ranges that are guaranteed to enclose an object’s location in the coordinate system.

The dimension table has the following columns:

1. Name: specifies the name of the dimension.
2. Datatype: identifies the datatype for the federate view of the dimension.
3. Dimension Upper Bound: specifies the upper bound for the dimension which meets the federation’s requirement for dimension subrange resolution. The value for this column shall be a positive integer.
4. Normalization Function: specifies the map from a subscription/update region’s bounding coordinates to non-negative integer subranges in the range \([0, \text{dimension upper bound})\).
5. Value When Unspecified: specifies a default range for the dimension that the RTI should use in overlap calculations if the dimension has been left unspecified when a federate creates a region that is subsequently used.

Figure 3.9 shows a partially completed example of the contents of the dimension table.
Figure 3.9 illustrates an example of the use of the dimension table. The first dimension, SodaFlavor, is an enumerated dimension, and its linearEnumerated normalization function would map Cola to [0..1), Orange to [1..2.), and RootBeer to [23). The next two dimensions, BarQuantity and WaiterId, are integer-based dimensions and would be mapped at uniform intervals across the ranges [0..25) and [0..20), respectively. Notice that the default Value When Unspecified for BarQuantity is a point range which is equivalent to [0..1).

### 3.1.7 Time Representation Table

Simulations provide a means of exercising system models over time. During a federation execution, time may play two roles. Federates may associate themselves with points on the HLA time axis, and they may associate some of their activities, such as updating an instance attribute’s value or sending an interaction, with points on the HLA time axis. These points on the HLA time axis are referred to as time stamps. As a federation execution progresses, federates may advance along the HLA time axis.

The HLA provides a variety of time management services to control the advancement of federates along the HLA time axis. These services allow multiple federates with differing internal time advancement strategies to interoperate in a meaningful way. For this reason, it is vital for all federation members to agree
upon how time stamps are represented across the federation, and document this agreement in the federation’s FOM. The time representation table is used for this purpose.

The specific strategy used to advance time in a simulation is driven by the simulation’s purpose. For example, faster-than-real time, event-stepped simulations are frequently used in analysis applications, while real-time, time-stepped simulations are often used in training applications. The strategy chosen for advancing time is an important consideration in the design of a simulation, as it affects both simulation performance and the ability of a simulation to interoperate with other simulations in a federation. For this reason, it is important to document how a simulation supports the advancement of time in a SOM.

During federation execution, time stamps are represented as instances of the time representation abstract datatype. This abstract datatype is provided to the RTI when a federate joins a federation execution so that an RTI will know how to use the time stamps. This table allows the datatype and semantics of the abstract datatype for time stamps to be explained in FOMs and SOMs.

It is also important to define the lookahead characteristics of both federates and federations. At the individual federate level, the specific means by which lookahead is supported may be a factor in assessing compatibility with other potential federates for a given federation application. For federations, it is important to document a common set of characteristics for how lookahead will be supported across the full federation. Thus, both SOMs and FOMs include representations of lookahead as well as time stamps. Representations of lookahead shall be drawn from non-negative types.

An abstract datatype for lookahead is supplied when a federate joins a federation execution. The lookahead entry in this table allows the datatype and semantics of this abstract datatype to be documented in FOMs and SOMs.

There are semantic differences between the way time is represented for the purpose of depicting time stamps versus calculating lookahead. When depicting time stamps, time can be considered to be an absolute value on a timeline (the HLA time axis), and thus time comparisons can be drawn to determine if one time stamp is greater than another. Lookahead, in contrast, represents a duration of time, which can be added to time stamps but is generally not used for comparison purposes.

The time representation table has the following columns:

1. **Category**: presents the two time-related values that are to be specified in
the table — time stamp and lookahead. \textit{Time Stamp} and \textit{Lookahead} correspond to the only two rows in this table.

2. \textit{Datatype}: identifies the datatype of the time value.
3. \textit{Semantics}: expands and describes the use of the datatype for time values.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|}
\hline
Category & Datatype & Semantics \\
\hline
Time Stamp & TimeType & Floating point value expressed in minutes \\
Lookahead & LAType & Floating point value expressed in minutes (non-negative) \\
\hline
\end{tabular}
\caption{Sample Time Representation Table}
\end{table}

\textbf{3.1.8 User-supplied Tag Table}

The HLA RTI provides a mechanism for federates to supply tags with certain of the HLA services. These tags can be used to provide additional coordination and control over these services. The user-supplied tag table provides a means of documenting federation agreements regarding the datatype to be used with these tags.

The first column (Category) presents the HLA service categories that are capable of accepting a user-supplied tag:
- \textit{Update/Reflect}: Updating and the corresponding reflection of instance attribute values.
- \textit{Send/Receive}: Sending and corresponding reception of an interaction.
- \textit{Delete/Remove}: Deletion and the corresponding removal of an object instance.
- \textit{Divestiture Request}: Negotiations involved in divesting ownership of instance attributes.
- \textit{Divestiture Completion}: Confirmation and completion of the negotiated
divestiture of ownership of instance attributes.

— Acquisition Request: Negotiations involved in acquiring ownership of instance attributes.

— Request Update: Requesting update of instance attribute values and the corresponding RTI direction to provide instance attribute values.

These seven service categories correspond to the only seven rows in this table. The second column (Datatype) identifies the datatype of the user-supplied tag for those categories of service that the federate or federations designate as providing a user-supplied tag. The third column (Semantics) expands and describes the use of the datatype for this user-supplied tag.

Sample User-supplied Tag Table

<table>
<thead>
<tr>
<th>Category</th>
<th>Datatype</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update/Reflect</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Send/Receive</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Delete/Remove</td>
<td>HLAASCIIstring</td>
<td>Reason for deletion</td>
</tr>
<tr>
<td>Divestiture Request</td>
<td>PriorityLevel</td>
<td>High value for immediate transfer</td>
</tr>
<tr>
<td>Divestiture Completion</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Acquisition Request</td>
<td>PriorityLevel</td>
<td>High value for immediate transfer</td>
</tr>
<tr>
<td>Request Update</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

Figure 3.11 User-supplied Tag Table
### 3.1.9 Synchronization Table

The HLA RTI provides a mechanism for federates to synchronize activities using a feature called *Synchronization Points*. The synchronization table provides a means for a federate to describe the synchronization points that it is capable of honoring and for a federation to document agreements regarding synchronization points to be used.

The first column (Label) contains a text string that defines the label associated with a synchronization point.

The second column (Tag Datatype) identifies the datatype of the user-supplied tag for those synchronization points that the federate or federation designate as providing a user-supplied tag.

The third column (Capability) indicates the level of interaction that a federate is capable of honoring. For FOMs, this column does not apply and shall contain “NA”. For SOMs, valid values for this column are:

- **Register**: The federate is capable of invoking HLA services to register the synchronization point.
- **Achieve**: The federate is capable of invoking HLA services to indicate achieving the synchronization point.
- **RegisterAchieve**: The federate is capable of both registering and achieving the synchronization point.
- **NoSynch**: The federate is capable of neither registering nor achieving the synchronization point.

The fourth column (Semantics) expands and describes the use of the synchronization point.

<table>
<thead>
<tr>
<th>Label</th>
<th>Tag Datatype</th>
<th>Capability</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

HLA Module 1, Part 3
### 3.1.10 Transportation type Table

The HLA RTI provides different mechanisms for the transportation of data (interactions and object instance attribute values) among federates. The transportation type table provides a means for a federate designer to describe the types of transportation that can be supported and for federation designers to document agreements regarding transportation of instance attributes and interactions.

Two transportation types, HLA\textit{reliable} and HLA\textit{bestEffort}, are required by the HLA and are described in detail in IEEE P1516.1. These types shall be provided by all RTIs. Other transportation types may be provided by specific RTIs.

The first column (Name) contains a text string that defines the name associated with a transportation type.

The second column (Description) describes the transportation type.
### Sample Transportation Type Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLA Reliable</td>
<td>Provides reliable delivery of data (in the sense that TCP/IP delivers its data reliably)</td>
</tr>
<tr>
<td>HLA best Effort</td>
<td>Makes an effort to deliver the data (in the sense that UDP provides effort delivery)</td>
</tr>
<tr>
<td>Lowest latency</td>
<td>Choose the delivery mechanism that results in the lowest latency from service initiation to callback invocation at the receiving federate</td>
</tr>
</tbody>
</table>

**Figure 3.13 Transportation Type Table**

#### 3.1.11 Switches Table

The HLA RTI performs actions on behalf of federates. Some of these actions may be enabled or disabled based on the capabilities of the federates or desires of the federation designers. These actions include automatically soliciting updates of instance attribute values when an object is newly discovered (controlled by the Auto Provide switch); and advising federates when certain events occur (controlled by the advisory switches). The switches table permits the specification of the initial setting of each switch, indicating whether or not the RTI should perform these actions. The functionality of these switches is described fully in IEEE P1516.1.

While the initial setting of each switch is specified in this table, the value of each switch may be changed during execution. The *Auto Provide* and *Convey Region Designator Sets* switches are controlled on a federation-wide basis; if any federate changes the switch settings, the change affects how the RTI interacts with all federates in the execution. The advisory switches, however, are controlled on a per-federate basis. Each federate in a federation has the same initial settings, as indicated in this table, but changes that an individual federate makes to an advisory switch only affect how the RTI interacts with that federate. Likewise, changes made to the *Service Reporting* switch are made on a per-federate basis. The RTI initially either reports service invocations for all federates or for none, as indicated in this table: changes made during a federation
execution only affect whether the RTI reports service invocations of a particular federate.

The switch settings are provided in a simple two-column table. The structure used to describe these settings is provided in Figure 3.14.

The first column (Switch) presents the name of the switch whose setting shall be provided in this table. The definitions of these switches are as follows:

- **Auto Provide**: determines whether the RTI should automatically solicit updates from instance attribute owners when an object is discovered.
- **Convey Region Designator sets**: determines whether the RTI should provide the optional *Sent Region Set* argument with invocations of *Reflect Attribute Values* and *Receive Interaction*.
- **Attribute Scope Advisory**: determines whether the RTI should advise federates when attributes of an object instance come into or go out of scope.
- **Attribute Relevance Advisory**: determines whether the RTI should advise federates about whether they should provide attribute value updates for the value of an attribute of an object instance; the RTI bases this advisory on whether the value of the instance attribute is required by other federates.
- **Object Class Relevance Advisory**: determines whether the RTI should advise federates about whether they should register instances of an object class; the RTI bases this advisory on whether other federates have expressed an interest in attribute(s) of the object class.
- **Interaction Relevance Advisory**: determines whether the RTI should advise federates about whether they should send interactions of an interaction class; the RTI bases this advisory on whether other federates have expressed an interest in the interaction class.
- **Service Reporting**: determines whether the RTI should report invocations using MOM.

The second column (Setting) specifies the setting for the switch. For SOMs, these entries are optional; “NA” shall be entered for all rows where no value is appropriate. For FOMs, entries shall be provided for all rows; valid values are:

- **Enabled**: the switch is enabled, and the RTI should perform the action when appropriate
- **Disabled**: the switch is disabled, and the RTI should not perform the action
3.1.12 Datatype Tables

Several of the OMT tables provide columns for datatype specifications. The characteristics of these datatypes are specified in additional tables: Simple Datatype Table, Enumerated Datatype Table, Fixed Record Datatype Table, Array Datatype Table, and Variant Record Datatype Table. Refer to section 4.12 of the IEEE P1516.2 (pp. 34-53) for details.

3.1.13 Notes Table

Any entry within any of the OMT tables may be annotated with additional descriptive information outside of the immediate table structure. This feature permits users to associate explanatory information with table entries to facilitate effective use of the data.

To attach one or more notes to an OMT table entry, include a notes pointer in the appropriate table cell. In the tabular OMT format this notes pointer consists of a uniquely-identifying note label (or a series of comma-separated labels) preceded by an asterisk, and enclosed by brackets. The notes themselves are associated with the note label and included in this table. Observe that a single note may be referenced numerous times in OMT tables and that a single OMT
table entry may reference numerous notes.

The first column (Label) shall contain a label for every notes pointer referenced in the object model.

The second column (Semantics) shall contain the explanatory text that constitutes the note.

<table>
<thead>
<tr>
<th>Label</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Merit raises are not provided according to any regular time interval; they are provided on a supervisor's recommendation based on evidence of exceptional effort and performance</td>
</tr>
<tr>
<td>2</td>
<td>Years of service are a factor in any merit raise. This value is only changed on the anniversary of the employee's hire.</td>
</tr>
</tbody>
</table>

Figure 3.15 Notes Table

3.1.14 FOM/SOM Lexicon

The FOM/SOM Lexicon is included in the OMT to provide semantic information necessary to an understanding of the information contained in the OMT tables. The lexicon consists of tables of definitions of object classes, interaction classes, attributes and parameters. Figure 3.16 shows the simple templates used to describe this information.
3.2 OMT Data Interchange Format (DIF) and FOM Document Data (FDD)

The HLA OMT data interchange format (DIF) is a standard file exchange format used to store and transfer HLA FOMs and SOMs between FOM/SOM builders. DIF is built upon a common meta-model that represents the information needed to represent and manage object models created using the HLA OMT standard. A formal definition of the DIF in XML is contained in the IEEE P1516.2, Annex D. An OMT DIF SOM example and an OMT DIF FOM example can be found in the IEEE P1516.2, Annex E and Annex F, respectively.

The question of how to document a simulation model in a formal way that expresses all of the details needed to judge its capabilities has long been a major concern of simulation developers and users. For HLA simulations, at least, the use of DIF solves this problem since it tells a potential user of a federate or federation, or a developer of a new HLA component, exactly what to expect in terms of the external characteristics of a federate, or the nature of the data exchange within a federation.

The FDD is that data and information from a FOM document that is used by the Create Federation Execution service to initialize a federation execution. Note that the RTI 1.3-NG refers to a file containing the required FOM data as a FED file. This file is automatically produced by a tool for editing FOM data, as discussed in the next section.
To support the required HLA functionality, the FDD must contain, at a minimum, the data from the following tables in the FOM document:

1. Object class structure table
2. Interaction class structure table
3. Attribute table (transportation order, and available dimensions columns only)
4. Parameter table (transportation order, and available dimensions columns only)
5. Dimension table
6. Transportation type table
7. Switches table

### 3.3 Object Model Development Tool (OMDT)

At first sight, the development of FOMs and SOMs looks like a daunting task, but tools are available that automate and simplify this process. The key tool provided for this purpose for the RTI-NG 1.3 is the Object Model Development Tool (OMDT).

The OMDT, which is available free from the HLA web-site http://hla.dmso.mil, automates the process of constructing SOMs and FOMs. Once an object model has been constructed, the OMDT can generate the FDD file required by the RTI to execute the federation. Full details of the use of the OMDT can be found in the HLA OMDT User’s Guide which accompanies the tool software.

### Assignment

Use the *HLA Course Lab Notes* to help you make a FOM and a SOM for the initial version of *HelloWorld* used in Part 1 of this course. Generate a new FDD file using the OMDT and compare against the FDD file you have been using for *HelloWorld* in Parts 1 and 2.

### Suggested Readings
