IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) — Framework and Rules

Sponsor

Simulation Interoperability Standards Committee (SISC)
of the
IEEE Computer Society

Approved 21 September 2000

IEEE-SA Standards Board

Abstract: The High Level Architecture (HLA)—Framework and Rules is the capstone document for a family of related HLA standards. It defines the HLA, its components, and the rules that outline the responsibilities of HLA federates and federations to ensure a consistent implementation. Simulations are abstractions of the real world, and no one simulation can solve all of the functional needs for the modeling and simulation community. It is anticipated that technology advances will allow for new and different modeling and simulation (M&S) implementations within the framework of the HLA. The standards contained in this architecture are interrelated and need to be considered as a product set, as a change in one is likely to have an impact on the others. As such, the HLA is an integrated approach that has been developed to provide a common architecture for simulation.
Keywords: architecture, class attribute, federate, federation, federation execution, federation object model, framework, high level architecture, instance attribute, interaction class, joined federate, object class, object model template, rules, runtime infrastructure, simulation object model

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Print: ISBN 0-7381-2619-5 SH94882
PDF: ISBN 0-7381-2620-9 SS94882

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

Note: Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

IEEE is the sole entity that may authorize the use of certification marks, trademarks, or other designations to indicate compliance with the materials set forth herein.

Introduction

(This introduction is not part of IEEE Std 1516-2000, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules.)

This standard is the capstone document for a family of related standards that together describe a unified approach to constructing interoperable simulation systems.

The HLA provides a general framework within which simulation developers can structure and describe their simulation applications. Flexibility is the aim of the HLA. In particular, the HLA addresses two key issues: promoting interoperability between simulations and aiding the reuse of models in different contexts. Two main components are described within the set of products forming the HLA. The first is the Object Model Template (OMT), which forms a documentation standard describing the data used by a particular model, a necessary basis for reuse. The second component, the Federate Interface Specification, describes a generic communications interface that allows simulation models to be connected and coordinated, thus, addressing interoperability. Although the HLA is an architecture, not software, use of runtime infrastructure (RTI) software is required to support operations of a federation execution. The RTI software provides a set of services, as defined by the Federate Interface Specification, used by federates to coordinate operations and data exchange during a runtime execution.

Simulations are necessarily abstractions of the real world, and no one simulation design can meet the functional needs of the entire modeling and simulation community. However, in defining an overriding architecture, generic issues can be addressed. When doing so, it is essential that such an architecture encompass both differing computing environments and differing classes of simulations.

This standard document of the Framework and Rules is intended to provide some of the general philosophy behind the HLA, including guidance about how to design, use, and adhere to the HLA vision.

Participants

This standard was prepared by the HLA Working Group, sponsored by the Simulation Interoperability Standards Committee (SISC) of the IEEE Computer Society. When the 1516 working group approved this standard, it had the following membership:

Susan Symington, *Chair*
Katherine L. Morse, *Vice Chair*
Mikel Petty, *Secretary*

Stephen Bachinsky
Joe Batman
Brian Beebe
Keith Briggs
Jim Calvin
Richard M. Fujimoto
Deborah Fullford
Allison Griffin
James H. Hammond
James Ivers
Andreas Kemkes
John F. Kramer
Frederick S. Kuhl
Gary M. Lightner
Reed Little
Rodney Long
Margaret Loper
Robert R. Lutz
Jeffrey M. Nielsen
Marnie R. Salisbury
Randy Saunders
Roy Scrudder
David W. Seidel
Robert C. Turrell
Richard Weatherly
Annette Wilson

The working group acknowledges the following individuals who also contributed to the preparation of this standard:

Phillippe Annic
Tobin Bergen-Hill
Dominique Canazzi
Peter Hoare
Judith S. Dahmann
Mikael Karlsson
Michael Mazurek
J. Russell Noseworthy
Graham Shanks

The following members of the balloting committee voted on this standard:

Dale E. Anglin
Stephen Bachinsky
Michael R. Bachmann
David L. Barton
Brian Beebe
Robert P. Bennett
Christina L. Bouwens
Keith Briggs
Richard Briggs
Danny Cohen
Glenn Conrad
Mark Crooks
Dannie Cutts
Judith S. Dahmann
Steven Drake
Laura E. Feinerman
Jeff E. Fischer
Deborah Fullford
Richard M. Fujimoto
Brian F. Goldiez
Victor M. Gonzalez
James H. Hammond
Jack G. Harrington
Robert V. Head, Jr
Callie M. Hill
Ronald C. Hofer
James W. Hollenbach
Margaret M. Horst
Robert J. Howard
Ken Hunt
Kyle Isakson
James Ivers
Andreas Kemkes
John F. Kramer
Frederick S. Kuhl
Tom Lake
Gary M. Lightner
Paul R. Little
Reed Little
Margaret Loper
Robert R. Lutz
Jayne Lyons
Theodore F. Marz
Mark McAuliffe
David R. McKeeby
Duncan C. Miller
Katherine L. Morse
Thomas H. Mullins
Jeffrey M. Nielsen
Michael J. O'Connor
John Peng
Paul Perkinson
Hung Q. Phan
David R. Pratt
Paul M. Reeves
David Roberts
Marnie R. Salisbury
Randy Saunders
Roy Scrudder
David W. Seidel
Richard J. Severinghaus
Joseph F. Seward
Sean Sharp
Steven M. Sheasby
John W. Sheppard
Allen H. Skees
Col. Mark Smith
Susan D. Solick
Sandra Swearingen
Stephen J. Swenson
Susan F. Symington
Donald Theune
William V. Tucker
Grant R. Tudor
John A. Tufarolo
Robert C. Turrell
Andrew J. Ventre
William F. Waite
Charles E. Walters
Richard Weatherly
Chris C. Wertman
Annette Wilson
Robert Whittman
Doug Wood
Philomena M. Zimmerman
William H. Zimmerman

When the IEEE-SA Standards Board approved this standard on 21 September 2000, it had the following membership:

Donald N. Heirman, *Chair*
James T. Carlo, *Vice Chair*
Judith Gorman, *Secretary*

Satish K. Aggarwal
Mark D. Bowman
Gary R. Engmann
Harold E. Epstein
H. Landis Floyd
Jay Forster*
Howard M. Frazier
Ruben D. Garzon
James H. Gurney
Richard J. Holleman
Lowell G. Johnson
Robert J. Kennelly
Joseph L. Koepfinger*
Peter H. Lips
L. Bruce McClung
Daleep C. Mohla
James W. Moore
Robert F. Munzner
Ronald C. Petersen
Gerald H. Peterson
John B. Posey
Gary S. Robinson
Akio Tojo
Donald W. Zipse

*Member Emeritus

Also included is the following nonvoting IEEE-SA Standards Board liaison:

Alan Cookson, *NIST Representative*
Donald R. Volzka, *TAB Representative*

Jennifer McClain Longman
IEEE Standards Project Editor

Contents

IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules

1. Overview

1.1 Scope

This document provides an overview of the High Level Architecture (HLA), defines a family of related HLA documents, and defines the principles of HLA in terms of responsibilities that federates (simulations, supporting utilities, or interfaces to live systems) and federations (sets of federates working together) must uphold.

1.2 Purpose

This document describes the general principles defining the HLA and delineates the set of rules that apply to HLA federations and federates. Each rule is then described, and the rationale for its inclusion is provided.

Many different classes of simulations exist. Each class has changing application characteristics and needs to be flexibly supported to allow for interoperability and reuse across classes and to limit the need to maintain multiple interoperability approaches. The HLA is an integrated architecture, which has been developed to provide that common architecture. The related standards need to be considered as a set of products because changes in one are likely to have an impact on the others.

1.3 Relationship of HLA and object-oriented concepts

Although the HLA Object Model Template (OMT) is the standardized documentation structure for HLA object models, Federation Object Models (FOMs) and Simulation Object Models (SOMs) do not completely correspond to common definitions of object models in object-oriented (OO) analysis and design (OOAD) techniques. In the OOAD literature, an object model is described as an abstraction of a system developed for the purpose of fully understanding the system. To achieve this understanding, most OO techniques recommend defining several views of the system. For HLA object models, the intended scope of the system description is much narrower, focusing specifically on requirements and capabilities for federate information exchange. For SOMs, the intent is to describe the public interface of the federate in terms of an

identified set of supported HLA object classes and interaction classes. A more complete description of how a federate is designed and functions internally (for example, a traditional OO object model) should be provided via documentation resources other than the SOM. For FOMs, the intent is to describe information exchange that happens during a federation execution.

Differences between HLA and OOAD principles and concepts also appear at the individual object level. In the OOAD literature, objects are defined as software encapsulations of data and operations (methods). In the HLA, objects are defined entirely by the identifying characteristics (attributes), values of which are exchanged between federates during a federation execution. Any OO-related behaviors and operations that affect the values of HLA object attributes are kept resident in the federates. Although HLA class structures are driven by subscription requirements and FOM growth concerns, class structures in OO systems are typically driven by efficiency and maintainability concerns.

As discussed above, HLA object classes are described by the attributes that are defined for them. These are, in OO parlance, data members of the class. These attributes, which are abstract properties of HLA object classes, are referred to as class attributes. HLA object instances are spawned via an HLA service using an HLA object class as a template. Each attribute contained by an HLA object instance is called an instance attribute.

OO objects interact via message passing, in which one OO object invokes an operation provided by another OO object. HLA objects do not directly interact. It is the federates that interact, via HLA services, by updating instance attribute values or sending interactions. Also, responsibility for updating the instance attributes associated with an HLA object instance can be distributed among different federates in a federation (effectively distributing responsibility for maintaining the HLA object instance's state across the federation), whereas OO objects encapsulate state locally and associate update responsibilities with operations that are closely tied to the object's implementation in an OO programming language.

In addition to the stated semantic variations in shared terminology, other differences may also exist. Precise definitions of all HLA terms can be found in Clause 3.

2. References

This standard shall be used in conjunction with the following publications. When the following specifications are superseded by an approved revision, the revision shall apply:

IEEE 1516.1-2000, IEEE Standard for Modeling and Simulation (M&S) High-Level Architecture (HLA)—Federate Interface Specification.[1]

IEEE 1516.2-2000, IEEE Standard for Modeling and Simulation (M&S) High-Level Architecture (HLA)—Object Model Template (OMT) Specification.

3. Definitions, abbreviations and acronyms

3.1 Definitions

For the purposes of this standard, the following terms and definitions apply. *The Authoritative Dictionary of IEEE Standards Terms* [B1][2] should be referenced for terms not defined in this clause.

[1]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA (http://standards.ieee.org/).

[2]The numbers in brackets correspond to those of the bibliography in Annex B.

3.1.1 accuracy: The measure of the maximum deviation of an attribute or a parameter value in the simulation or federation from reality or some other chosen standard or referent.

3.1.2 active subscription: A request to the runtime infrastructure (RTI) for the kinds of data (class attributes as well as interactions) that the joined federate is currently interested in receiving. The RTI uses this data along with publication data received from other joined federates to support services, such as

a) Start/Stop Registration
b) Turn On/Off Updates
c) Turn On/Off Interactions

The RTI also uses the subscription (and publication) data to determine how to route data to the joined federates that require that data.

3.1.3 attribute: A named characteristic of an object class or object instance. *See also*: **class attribute** and **instance attribute**.

3.1.4 attribute ownership: The property of an instance attribute that gives a joined federate the capability to supply values for that instance attribute to its federation execution. *See also*: **instance attribute**.

3.1.5 available attributes: The set of declared attributes of an object class in union with the set of inherited attributes of that object class.

3.1.6 available dimensions:

a) Pertaining to an attribute: the dimensions associated with the class attribute in the Federation Object Model (FOM). The available dimensions of an instance attribute are the available dimensions of the corresponding class attribute.
b) Pertaining to an interaction class: the dimensions associated with that interaction class in the FOM. The available dimensions of a sent interaction are the available dimensions of the interaction class specified in the *Send Interaction With Regions* service invocation.

NOTE—See 9.1 in IEEE Std 1516.1-2000.

3.1.7 available parameters: The set of declared parameters of an interaction class in union with the set of inherited parameters of that interaction class.

3.1.8 candidate discovery class: The registered class of an object instance, if subscribed. If the registered class of an object instance is not subscribed, the closest superclass of the registered class of the object instance to which the joined federate is subscribed. The term candidate discovery class pertains to object instances only.

3.1.9 candidate received class: The sent class of an interaction, if subscribed. If the sent class of an interaction is not subscribed, the closest superclass of the sent class of the interaction to which the joined federate is subscribed. The term candidate received class pertains to interactions only.

3.1.10 class: A description of a group of items with similar properties, common behavior, common relationships, and common semantics.

3.1.11 class attribute: A named characteristic of an object class denoted by a pair composed of an object class designator and an attribute designator.

3.1.12 class hierarchy: A specification of a class–subclass or "is–a" relationship between classes in a given domain.

3.1.13 coadunate: To attach an object instance handle (and possibly an object instance name) to an object instance.

3.1.14 collection: A set in which an element may occur multiple times (this corresponds to the mathematical notion of a bag).

3.1.15 collection of pairs: A group of pairs in which multiple pairs may have the same first component and a given pair may occur multiple times.

3.1.16 compliant object model: A High Level Architecture (HLA) Federation Object Model (FOM) or Simulation Object Model (SOM) that fully conforms with all of the rules and constraints specified in Object Model Template (OMT).

NOTE—See IEEE Std 1516.2-2000.

3.1.17 constrained set of pairs: A group of pairs in which no two pairs have the same first component (this corresponds to the mathematical notion of a function). An example would be a group of instance attribute and value pairs; each instance attribute may have at most one value associated with it.

3.1.18 corresponding class attribute of an instance attribute: The class attribute that, from the perspective of a given joined federate, is the class attribute of the joined federate's known class for the object instance containing the instance attribute that has the same attribute designator as the instance attribute.

3.1.19 corresponding instance attributes of a class attribute: The instance attributes that, from the perspective of a given joined federate, are

a) Unowned instance attributes of object instances that have a known class at the joined federate equal to the object class of the class attribute and that have the same attribute designator as the class attribute, or

b) Instance attributes owned by the joined federate that belong to object instances that have a known class at the owning federate equal to the object class of the class attribute and that have the same attribute designator as the class attribute.

3.1.20 datatype: A representation convention for a data element establishing its format, resolution, cardinality, and ordinality.

3.1.21 declared attributes: The set of class attributes of a particular object class that are listed in the Federation Object Model (FOM) as being associated with that object class in the object class hierarchy tree.

3.1.22 declared parameters: The set of parameters of a particular interaction class that are listed in the Federation Object Model (FOM) as being associated with that interaction class in the interaction class hierarchy tree.

3.1.23 default range: A range lower bound and a range upper bound, defined in the Federation Object Model Document Data (FDD) and specified in terms of [0, the dimension's upper bound), for a dimension.

3.1.24 default region: A multidimensional region provided by the runtime infrastructure (RTI) that is composed of one range for each dimension found in the Federation Object Model Document Data (FDD). The bounds of each of these ranges are [0, the range's dimension's upper bound). There is no way for a federate to refer to the default region.

NOTE—See 9.1 in IEEE Std 1516.1-2000.

3.1.25 designator: Some arguments (mostly identifiers) to services may have different views (implementations), depending on a particular programming language, application programmer's interface (API). For clarity, service descriptions refer to a generic view of the arguments, known as a *designator*.

3.1.26 dimension: A named interval. The interval is defined by an ordered pair of values, the first being the dimension lower bound and the second being the dimension upper bound. A runtime infrastructure (RTI) provides a predefined interval, whose lower and upper bounds are fixed as [0, the dimension's upperbound as specified in the Federation Object Model Document Data (FDD)]. This interval provides a single basis for communication of all dimension-related data with an RTI. All normalized intervals communicated to the RTI will be subsets of this interval.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.1.27 discover: To receive an invocation of the *Discover Object Instance †*[3] service for a particular object instance.

NOTE—See 6.5 of IEEE Std 1516.1-2000.

3.1.28 discovered class: The class that was an object instance's candidate discovery class at a joined federate when that object instance was discovered by that joined federate. *See also*: **candidate discovery class.**

NOTE—See 5.1.2 in IEEE Std 1516.1-2000.

3.1.29 exception: Notification of any irregularity that may occur during a service invocation.

3.1.30 federate: An application that may be or is currently coupled with other software applications under a Federation Object Model Document Data (FDD) and a runtime infrastructure (RTI). *See also*: **federate application** and **joined federate.**

3.1.31 federate application: An application that supports the High Level Architecture (HLA) interface to a runtime infrastructure (RTI) and that is capable of joining a federation execution. A federate application may join the same federation execution multiple times or may join multiple federation executions. However, each time a federate application joins a federation execution, it is creating a new joined federate. *See also*: **joined federate**.

3.1.32 federation: A named set of federate applications and a common Federation Object Model that are used as a whole to achieve some specific objective.

3.1.33 federation execution: The actual operation, over time, of a set of joined federates that are interconnected by a runtime infrastructure (RTI).

3.1.34 Federation Object Model (FOM): A specification defining the information exchanged at runtime to achieve a given set of federation objectives. This includes object classes, object class attributes, interaction classes, interaction parameters, and other relevant information.

3.1.35 Federation Object Model (FOM) Document Data (FDD): The data and information in an FOM document that is used by the *Create Federation Execution* service to initialize a newly created federation execution

NOTE—See IEEE Std 1516.2-2000.

[3]All RTI initiated services are denoted with a † (printer's dagger) after the service name.

3.1.36 handle: An identifier originated/created by the runtime infrastructure (RTI) that is federation execution-wide unique and unpredictable.

3.1.37 High Level Architecture (HLA) time axis: A totally ordered sequence of values in which each value typically represents an HLA instant of time in the physical system being modeled. For any two points T1 and T2 on the time axis, if T1 < T2, T1 represents an instant of time that occurs before the instant represented by T2.

3.1.38 in scope: Of or pertaining to an instance attribute of an object instance for which

a) The object instance is known to the joined federate

b) The instance attribute is owned by another joined federate, and

c) Either the instance attribute's corresponding class attribute is a

   1) Subscribed attribute of the known class of the object instance, or

   2) Subscribed attribute of the known class of the object instance with regions, and the update region set of the instance attribute at the owning joined federate overlaps the subscription region set of the instance attribute's corresponding class attribute at the known class of the instance attribute at the subscribing joined federate

      NOTE—See 6.1 of IEEE Std 1516.1-2000.

3.1.39 inherited attribute: A class attribute of an object class that was declared in a superclass of that object class in the object class hierarchy tree defined in the Federation Object Model (FOM).

3.1.40 inherited parameter: An interaction parameter that was declared in a superclass of that interaction class in the interaction class hierarchy tree defined in the Federation Object Model (FOM).

3.1.41 instance attribute: A named characteristic of an object instance denoted by a pair composed of the object instance designator and the attribute designator.

3.1.42 interaction: An explicit action taken by a federate that may have some effect or impact on another federate within a federation execution.

3.1.43 interaction class: A template for a set of characteristics that is common to a group of interactions. These characteristics correspond to the parameters that individual federates may associate with interactions.

3.1.44 interaction parameters: The information associated with an interaction that a federate potentially affected by the interaction may receive to calculate the effects of that interaction on its current state.

3.1.45 joined federate: A member of a federation execution, actualized by a federate application invoking the *Join Federation Execution* service as prescribed in IEEE Std 1516.1-2000. *See also*: **federate application**.

3.1.46 known class:

a) An object instance's registered class if the joined federate knows about the object instance as a result of having registered it, or

b) An object instance's discovered class if the joined federate knows about the object instance as a result of having discovered it.

3.1.47 known object instance: An object instance that a given joined federate has either registered or discovered, and one that the joined federate has not subsequently deleted (globally or locally) or was notified to remove. *See also*: **register** and **discover**.

3.1.48 logical time: A federate's current point on the High Level Architecture (HLA) time axis. Federates making use of the time management services follow restrictions on what time stamps can be sent in time stamp order (TSO) messages (relative to their logical time) to ensure that federates receiving those messages receive them in TSO.

3.1.49 lookahead: Lookahead is a nonnegative value that establishes a lower value on the time stamps that can be sent in time stamp order (TSO) messages by time-regulating joined federates. Once established, a joined federate's lookahead value may only be changed using the *Modify Lookahead* service. Each time-regulating joined federate must provide a lookahead value when becoming time-regulating.

3.1.50 Management Object Model (MOM): A group of predefined High Level Architecture (HLA) constructs (object and interaction classes) that provide the following:

a) Access to federation execution operating information
b) Insight into the operations of joined federates and the runtime infrastructure (RTI), and
c) Control of the functioning of the RTI, the federation execution, and the individual joined federates

3.1.51 message: A change of object instance attribute value, an interaction, or a deletion of an existing object instance, often associated with a particular point on the High Level Architecture (HLA) time axis, as denoted by the associated time stamp.

3.1.52 object class: A fundamental element of a conceptual representation for a federate that reflects the real world at levels of abstraction and resolution appropriate for federate interoperability. A template for a set of characteristics that is common to a group of object instances. These characteristics correspond to the class attributes that individual federates may publish and to which other federates may subscribe.

3.1.53 object instance: A unique instantiation of an object class that is independent of all other instances of that object class. At any point during a federation execution, the state of a High Level Architecture (HLA) object instance is defined as the collection of the values of all its instance attributes.

3.1.54 object model: A system specification defined primarily by class characteristics and relationships. The High Level Architecture (HLA) idea of an object model is similar in many ways, but not identical, to the common idea of an object model in object-oriented literature.

3.1.55 object model framework: The rules and terminology used to describe High Level Architecture (HLA) object models.

3.1.56 order type: A runtime infrastructure (RTI) provided means of ordering messages originating from multiple joined federates that are delivered to a single joined federate. Different categories of service are defined with different characteristics regarding whether and how an RTI orders messages that are to be delivered to a joined federate.

3.1.57 out of scope: Of or pertaining to an instance attribute of an object instance for which one or more of the following is not true:

a) The object instance is known to the joined federate
b) The instance attribute is owned by another joined federate, and
c) Either the instance attribute's corresponding class attribute is a

1) Subscribed attribute of the known class of the object instance, or

2) Subscribed attribute of the known class of the object instance with regions, and the update region set of the instance attribute at the owning joined federate overlaps the subscription region set of the instance attribute's corresponding class attribute at the known class of the instance attribute at the subscribing joined federate.

NOTE—See 6.1 of IEEE Std 1516.1-2000.

3.1.58 overlap:

a) Pertaining to region sets: Two region sets overlap if there is a region in each set, such that the two regions overlap.

b) Pertaining to regions: If two regions have at least one dimension in common, they overlap if all ranges of dimensions that are contained in both regions overlap pairwise. If two regions do not have any dimensions in common, they do not overlap.

c) Pertaining to ranges: Two ranges A = $[a_{lower}, a_{upper})$ and B = $[b_{lower}, b_{upper})$ overlap, if and only if either $a_{lower} = b_{lower}$ or $(a_{lower} < b_{lower} < a_{upper})$.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.1.59 owned: Pertaining to the relationship between an instance attribute and the joined federate that has the unique right to update that instance attribute's value.

3.1.60 owned instance attribute: An instance attribute that is explicitly modeled by the owning joined federate. A joined federate that owns an instance attribute has the unique responsibility to provide values for that instance attribute to the federation, through the runtime infrastructure (RTI), as documented in the Federation Object Model Document Data (FDD).

3.1.61 pair: A grouping of two related elements (a first component and a second component), the combination of which is treated as an entity. An example of a pair would be an instance attribute grouped with its current value.

3.1.62 parameter: A named characteristic of an interaction.

3.1.63 passel: A group of attribute handle/value pairs from an *Update Attribute Values* service invocation that are delivered together via a *Reflect Attribute Values †* service invocation. All pairs within the passel have the same user-supplied tag, sent message order type, transportation type, receive message order type, time stamp (if present), and set of sent region designators (if present). A passel is a message.

3.1.64 passive subscription: A request to the runtime infrastructure (RTI) for the kinds of data (object classes and attributes as well as interactions) that the joined federate is currently interested in receiving, but unlike an active subscription, this information is not used by the RTI to arrange for data to be delivered, nor is it used to tell publishing joined federates that another joined federate is subscribing to that data (by way of *Start/Stop Registration*, *Turn On/Off Updates,* or *Turn On/Off Interactions* service invocations). This form of subscription is provided to support certain types of logger joined federates.

3.1.65 promoted: Pertaining to an object instance, as known by a particular joined federate, that has a discovered class that is a superclass of its registered class.

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

3.1.66 published:

a) Pertaining to an object class such that, from the perspective of a given joined federate, there is at least one available attribute of the object class that was an argument to a *Publish Object Class Attributes* service invocation that was not subsequently unpublished via the *Unpublish Object Class Attribut*es service.

NOTE—See 5.1.2 of IEEE Std 1516.1-2000.

b) Pertaining to an interaction class that, from the perspective of a given joined federate, was an argument to a *Publish Interaction Class* service invocation that was not subsequently followed by an *Unpublish Interaction Class* service invocation for that interaction class.

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

3.1.67 published attributes of an object class: The class attributes that have been arguments to *Publish Object Class Attributes* service invocations by a given joined federate for that object class that have not subsequently been unpublished (either individually or by unpublishing the whole object class), and possibly the **HLAprivilegeToDeleteObject** attribute for that object class.

NOTE—See 5.1.2 of IEEE Std 1516.1-2000.

3.1.68 range: A subset of a dimension, defined by an ordered pair of values, the first being the range lower bound and the second being the range upper bound. This pair of values defines a semi-open interval [range lower bound, range upper bound) (i.e., the range lower bound is the smallest member of the interval, and the range upper bound is just greater than any member of the interval).

NOTE—See 9.1.1 of IEEE Std 1516.1-2000.

3.1.69 range lower bound: The first component of the ordered pair of values that is part of a range.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.1.70 range upper bound: The second component of the ordered pair of values that is part of a range.

NOTE—See IEEE 9.1 of Std 1516.1-2000.

3.1.71 receive order (RO): A characteristic of no ordering guarantee for messages. Messages that are received as RO messages will be received in an arbitrary order by the respective joined federate. A time stamp value will be provided with the message if one was specified when the message was sent, but that time stamp has no bearing on message receipt order.

3.1.72 received class: The class that was an interaction's candidate received class at the joined federate when that interaction was received at that joined federate via an invocation of the *Receive Interaction †* service.

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

3.1.73 received parameters: The set of parameters received when the *Receive Interaction †* service is invoked. These parameters consist of the subset of the sent parameters of an interaction that are available parameters of the interaction's received class.

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

3.1.74 reflect: Receive new values for one or more instance attributes via invocation of the *Reflect Attribute Values †* service.

NOTE—See 6.7 of IEEE Std 1516.1-2000.

3.1.75 reflected instance attribute: An instance attribute that is represented but not explicitly modeled in a joined federate. The reflecting joined federate accepts new values of the reflected instance attribute as they are produced by some other federation member and provided to it by the runtime infrastructure (RTI), via the *Reflect Attribute Values †* service.

3.1.76 region: A generic term that refers to either a region specification or a region realization. If not using Data Distribution Management (DDM), region arguments may be ignored.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.1.77 region realization: A region specification (set of ranges) that is associated with an instance attribute for update, with a sent interaction, or with a class attribute or interaction class for subscription. Region realizations are created from region specifications via the *Commit Region Modifications* (only when modifying a region specification from which region realizations are already derived), *Register Object Instance With Regions*, *Associate Regions for Updates*, *Subscribe Object Class Attributes With Regions*, *Subscribe Interaction Class With Regions*, *Send Interaction With Regions*, or *Request Attribute Value Update With Regions* services.

NOTE—See 9.1.1 of IEEE Std 1516.1-2000.

3.1.78 region specification: A set of ranges. Region specifications are created using the *Create Region* service, and a runtime infrastructure (RTI) is notified of changes to a region specification using the *Commit Region Modifications* service.

NOTE—See 9.1.1 of IEEE Std 1516.1-2000.

3.1.79 register: To invoke the *Register Object Instance* or the *Register Object Instance With Regions* service to create a unique object instance designator.

NOTE—See 6.4 of IEEE Std 1516.1-2000.

3.1.80 registered class: The object class that was an argument to the *Register Object Instance* or the *Register Object Instance With Regions* service invocation that resulted in the creation of the object instance designator for a given object instance.

3.1.81 resolution: The smallest resolvable value separating attribute or parameter values that can be discriminated. Resolution may vary with magnitude for certain data types.

3.1.82 retraction: An action performed by a federate to unschedule a previously scheduled message. Message retraction may be visible to the federate to whom the scheduled message was to be delivered. Retraction is widely used in classic event-oriented discrete event simulations to model behaviors such as preemption and interrupts.

3.1.83 runtime infrastructure (RTI) initialization data (RID): RTI vendor-specific information needed to run an RTI. If required, an RID is supplied when an RTI is initialized.

3.1.84 runtime infrastructure (RTI): The software that provides common interface services during a High Level Architecture (HLA) federation execution for synchronization and data exchange.

3.1.85 sent class: The interaction class that was an argument to the *Send Interaction* or *Send Interaction With Regions* service invocation that initiated the sending of a given interaction.

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

3.1.86 sent interaction: A specific interaction that is transmitted by a joined federate via the *Send Interaction* or *Send Interaction With Regions* service and received by other joined federates in the federation execution via the *Receive Interaction †* service.

3.1.87 sent parameters: The parameters that were arguments to the *Send Interaction* or *Send Interaction With Regions* service invocation for a given interaction.

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

3.1.88 set: A group of elements in which each element occurs at most once (this corresponds to the mathematical notion of sets). An example of a set would be a group of class attributes, each of which belongs to the same object class.

3.1.89 Simulation Object Model (SOM): A specification of the types of information that an individual federate could provide to High Level Architecture (HLA) federations as well as the information that an individual federate can 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.

3.1.90 specified dimensions: The dimensions that are explicitly provided when the region specification is created or modified.

NOTE—See 9.1.2 of IEEE Std 1516.1-2000.

3.1.91 stop publishing: To take action that results in a class attribute that had been a published attribute of a class no longer being a published attribute of that class.

3.1.92 subclass: A class that adds additional detail to (specializes) another more generic class (superclass). A subclass, by inheriting the properties from its parent class (closest superclass), also inherits the properties of all superclasses of its parent as well.

3.1.93 subscribed:

a) Pertaining to an object class for which, from the perspective of a given joined federate, there are subscribed attributes of that class or subscribed attributes of that class with regions, for some region. *See also*: **subscribed attributes of an object class** and **subscribed attributes of an object class with regions**.

b) Pertaining to an interaction class that is a subscribed interaction class or a subscribed interaction class with regions, for some region. *See also*: **subscribed interaction class** and **subscribed interaction class with regions**.

3.1.94 subscribed attributes of an object class: The class attributes that have been arguments to *Subscribe Object Class Attributes* service invocations by a given joined federate for a given object class that have not subsequently been unsubscribed, either individually or by unsubscribing the whole object class.

NOTE—See 5.1.2 and 5.6 of IEEE Std 1516.1-2000.

3.1.95 subscribed attributes of an object class with regions: The class attributes that have been arguments to *Subscribe Object Class Attributes With Regions* service invocations by a given joined federate for a given object class and a given region, assuming the joined federate did not subsequently invoke the *Unsubscribe Object Class Attributes With Regions* service for that object class and region.

NOTE—See 9.8 of IEEE Std 1516.1-2000.

3.1.96 subscribed interaction class: Pertaining to an interaction class that, from the perspective of a given joined federate, was an argument to a *Subscribe Interaction Class* or *Subscribe Interaction Class With Regions* service invocation that was not subsequently followed by an *Unsubscribe Interaction Class* or *Unsubscribe Interaction Class With Regions* service invocation for that interaction class.

NOTE—See 5.1.3 and 5.8 of IEEE Std 1516.1-2000.

3.1.97 subscribed interaction class with regions: Pertaining to an interaction class and a region that, from the perspective of a given joined federate, were arguments to a *Subscribe Interaction Class With Regions* service invocation that was not subsequently followed by an *Unsubscribe Interaction Class With Regions* service invocation for that interaction class and that region.

NOTE—See 9.10 of IEEE Std 1516.1-2000.

3.1.98 subscription region set: A set of regions used for subscription of a class attribute or used for subscription of an interaction class. *See also*: **used for subscription of a class attribute** and **used for subscription of an interaction class**.

3.1.99 superclass: A class that generalizes a set of properties that may be inherited by more refined (i.e., detailed) versions of the class. In High Level Architecture (HLA) applications, a class may have at most one immediate superclass (i.e., can only inherit from a single class at the next highest level of the class hierarchy).

3.1.100 synchronization point: A logical point in the sequence of a federation execution that all joined federates forming a synchronization set for that point attempt to reach and, if they are successful, thereby synchronize their respective processing at that point.

3.1.101 time advancing state: A joined federate may advance its logical time only by requesting a time advancement from the runtime infrastructure (RTI) via one of the following services:

a) Time Advance Request
b) Time Advance Request Available
c) Next Message Request
d) Next Message Request Available
e) Flush Queue Request

The joined federate's logical time will not actually be advanced until the RTI responds with a *Time Advance Grant †* service invocation at that joined federate. During the interval between a request to advance its logical time and the corresponding grant, the joined federate is in the time advancing state.

3.1.102 time-constrained federate: A joined federate that may receive time stamp order (TSO) messages and whose time advances are constrained by other joined federates within a federation execution.

NOTE—See 8.1 of IEEE Std 1516.1-2000.

3.1.103 time-regulating federate: A joined federate that may send time stamp order (TSO) messages and that constrains the time advances of other joined federates within a federation execution.

NOTE—See 8.1 of IEEE Std 1516.1-2000.

3.1.104 time management: A collection of High Level Architecture (HLA) services that support controlled message ordering and delivery to the cooperating joined federates within a federation execution in a way that is consistent with federation requirements.

3.1.105 time stamp (of message or save): The value of the time stamp argument provided to the relevant service invocation.

3.1.106 time stamp order (TSO): An ordering of messages provided by a runtime infrastructure (RTI) for joined federates making use of time management services and messages containing time stamps. Messages having different time stamps are said to be delivered in TSO if for any two messages M1 and M2 (time stamped with T1 and T2, respectively) that are delivered to a single joined federate where T1 < T2, then M1

is delivered before M2. Messages having the same time stamp will be delivered in an arbitrary order (i.e., no tie-breaking mechanism is provided by an RTI).

3.1.107 transportation type: A runtime infrastructure (RTI) provided means of transmitting messages between joined federates. Different categories of service are defined with different characteristics such as reliability of delivery and message latency.

3.1.108 unspecified dimensions: The available dimensions of a class attribute, instance attribute, interaction class, or sent interaction less the specified dimensions of the region specification from which the region realization is derived.

NOTE—See 9.1.3 of IEEE Std 1516.1-2000.

3.1.109 update: Invoke the *Update Attribute Values* service for one or more instance attributes.

NOTE—See 6.6 of IEEE Std 1516.1-2000.

3.1.110 update region set: A set of regions used for sending interactions or used for update of instance attributes. *See also*: **used for sending** and **used for update**.

3.1.111 used for sending:

a) Pertaining to a region that, along with the specified interaction class designator, is being used as an argument in the *Send Interaction With Regions* service.

b) Pertaining to the default region when the specified interaction class designator is being used as an argument in the *Send Interaction* service.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.1.112 used for subscription of a class attribute:

a) Pertaining to a region, an object class, and a class attribute for which the class attribute is a subscribed attribute of the object class with that region. *See also*: **subscribed attributes of an object class with regions.**

NOTE—See 9.1 of IEEE Std 1516.1-2000.

b) Pertaining to the default region when the specified class attribute is a subscribed attribute of the specified class. *See also*: **subscribed attributes of an object class with regions.**

NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.1.113 used for subscription of an interaction class:

a) Pertaining to a region and an interaction class for which the interaction class is a subscribed interaction class with regions. *See also*: **subscribed interaction class**.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

b) Pertaining to the default region when the specified interaction class is a subscribed interaction class. *See also*: **subscribed interaction class**.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.1.114 used for update:

a) Pertaining to a region that, along with the specified object instance and attribute designators, has been used as an argument in either the *Register Object Instance With Regions* service or the

Associate Regions For Updates service; and the region has not subsequently been used along with the specified object instance designator as an argument in the *Unassociate Regions For Updates* service; nor has the joined federate subsequently lost ownership of the specified instance attribute(s).

NOTE—See 9.1 of IEEE Std 1516.1-2000.

b) Pertaining to the default region when the specified instance attribute(s) is not currently used for update with any other region.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

3.2 Abbreviations and acronyms

The following abbreviations and acronyms pertain to this standard.

API	application programmer's interface
BNF	Backus Naur Form
DDM	Data Distribution Management
DIF	data interchange format
DM	Declaration Management
DTD	Document Type Declaration
FDD	FOM Document Data
FOM	Federation Object Model
FQR	*Flush Queue Request*
GALT	Greatest Available Logical Time
HLA	High Level Architecture
LITS	Least Incoming Time Stamp
MOM	Management Object Model
M&S	modeling and simulation
NA	not applicable
NMR	*Next Message Request*
NMRA	*Next Message Request Available*
OMT	Object Model Template
OO	object oriented
OOAD	object-oriented analysis and design
POC	point of contact
RID	RTI initialization data
RO	receive order
RTI	runtime infrastructure
SISC	Simulation Interoperability Standards Committee
SOM	Simulation Object Model
TAR	*Time Advance Request*
TARA	*Time Advance Request Available*
TRADT	time representation abstract datatype
TSO	time stamp order
XML	eXtensible Markup Language

4. Summary of the HLA rules

The rules for federations are as follows:

1) Federations shall have an HLA FOM, documented in accordance with the HLA OMT.
2) In a federation, all simulation-associated object instance representation shall be in the federates, not in the RTI.
3) During a federation execution, all exchange of FOM data among joined federates shall occur via the RTI.
4) During a federation execution, joined federates shall interact with the RTI in accordance with the HLA interface specification.
5) During a federation execution, an instance attribute shall be owned by at most one joined federate at any given time.

The rules for federates are as follows:

6) Federates shall have an HLA SOM, documented in accordance with the HLA OMT.
7) Federates shall be able to update and/or reflect any instance attributes and send and/or receive interactions, as specified in their SOMs.
8) Federates shall be able to transfer and/or accept ownership of instance attributes dynamically during a federation execution, as specified in their SOMs.
9) Federates shall be able to vary the conditions (e.g., thresholds) under which they provide updates of instance attributes, as specified in their SOMs.
10) Federates shall be able to manage local time in a way that will allow them to coordinate data exchange with other members of a federation.

5. Federation rules

This clause describes the five rules that apply to HLA federations. Rationale for the inclusion of each rule is located in Annex A.

5.1 Rule 1

Federations shall have an HLA FOM, documented in accordance with the HLA OMT.

All data to be exchanged in accordance with the HLA shall be documented in a FOM. A FOM shall document the agreement among federates on data to be exchanged using the HLA services during federation execution and the minimal set of conditions of the data exchange (e.g., updates to be sent when changes exceed a certain value). As such, a FOM is an essential element in defining a federation. The HLA does not prescribe which data are included in a FOM (this is the responsibility of the federation user and developer). FOMs shall be documented in the format prescribed in IEEE Std 1516.2-2000 to support reuse of a FOM by new users for their purposes.

5.2 Rule 2

In a federation, all simulation-associated object instance representation shall be in the federates, not in the RTI.

In the HLA, the responsibility for maintaining the values of HLA object instance attributes shall take place in the joined federate. In an HLA federation, all joined federate-associated instance attributes shall be owned by federates, not by the RTI. However, the RTI may own instance attributes associated with the federation Management Object Model. The RTI may use data about instance attributes and interactions to support RTI services (e.g., Declaration Management), but these data are merely used by the RTI, not changed.

5.3 Rule 3

During a federation execution, all exchange of FOM data among joined federates shall occur via the RTI.

The HLA federate interface specification (see IEEE Std 1516.1-2000) specifies a set of interfaces to services in the RTI to support coordinated exchange of instance attribute values and interactions in accordance with a federation's FOM. Under the HLA, intercommunication of FOM data among joined federates participating in a given federation execution shall be executed by the exchange of data via the RTI services. Based on the FOM, joined federates shall identify to the RTI what information they will provide and require, along with instance attribute and interaction data corresponding to the changing state of object instances in the joined federate. The RTI shall then provide the coordination, synchronization, and data exchange among the joined federates to permit a coherent execution of the federation.

Ensuring that the right data are provided at the right times and that the data are used in a substantively correct way shall be the responsibility of the joined federates. The RTI shall ensure that the data are delivered using joined federates in accordance with their declared requirements (which data, reliability of transport, event ordering, etc.) to provide a common view of shared data across a given federation execution, as specified in the FOM.

RTI services shall be used to ensure that the coordination needs of the distributed applications (federations) are met in a consistent way across all participants in a federation and over the life of a federation execution.

5.4 Rule 4

During a federation execution, joined federates shall interact with the RTI in accordance with the HLA interface specification.

The HLA provides a specification for a standard interface between the federate application and the RTI. Joined federates shall use this standard interface to access RTI services (see IEEE Std 1516.1-2000). The specification shall define how federate applications interact with the infrastructure. However, because the interface and the RTI will be used for a wide variety of applications requiring data exchange of diverse characteristics, the interface specification says nothing about the specific federate data to be exchanged over the interface.

5.5 Rule 5

During a federation execution, an instance attribute shall be owned by, at most, one joined federate at any given time.

The HLA allows for different joined federates to own different attributes of the same object instance (e.g., a simulation of an aircraft might own the location of the airborne sensor, whereas a sensor system model might own other instance attributes of the sensor). To ensure data coherency across the federation, at most, one joined federate may own any given instance attribute of an object instance at any given time. Joined federates may request that the ownership of instance attributes be acquired or divested, dynamically, during federation execution. Thus, ownership can be transferred, dynamically during execution, from one joined federate to another.

6. Federate rules

This clause describes the five rules that apply to HLA federates. Rationale for the inclusion of each rule is located in Annex A.

6.1 Rule 6

Federates shall have an HLA SOM, documented in accordance with the HLA OMT.

The HLA SOM shall include those object classes, class attributes, and interaction classes of the federate that can be made public in a federation. The HLA does not prescribe which data are included in the SOM; this shall be the responsibility of the federate developer. SOMs shall be documented in the format prescribed in IEEE Std 1516.2-2000.

6.2 Rule 7

Federates shall be able to update and/or reflect any instance attributes and send and/or receive interactions, as specified in their SOMs.

The HLA allows for joined federates to make internal object representations and interactions available for external use as part of federation executions. These capabilities for external interaction shall be documented in the SOM for the federate. If documented in the SOM, these federate capabilities shall include the obligation to export updated values of instance attributes that are calculated internally in the federate and the obligation to be able to exercise interactions represented externally (i.e., by other federates in a federation).

6.3 Rule 8

Federates shall be able to transfer and/or accept ownership of instance attributes dynamically during a federation execution, as specified in their SOMs.

The HLA allows ownership of instance attributes of an object instance to be transferred dynamically during a federation execution. The instance attributes of a federate that can be either owned or reflected, and whose ownership can be dynamically acquired or divested during execution, shall be documented in the SOM for that federate.

6.4 Rule 9

Federates shall be able to vary the conditions (e.g., thresholds) under which they provide updates of instance attributes, as specified in their SOMs.

The HLA permits federates to own (i.e., provides the privilege to produce updated values for) instance attributes of object instances represented in the federate and to then make those values available to other federates through the RTI. Different federations may specify different conditions under which instance attributes will be updated [e.g., at some specified rate, or when the amount of change in value exceeds a specified threshold (such as altitude changes of more than 1000 ft, etc.)]. The conditions applicable to the update of specific instance attributes owned by a federate shall be documented in the SOM for that federate.

6.5 Rule 10

Federates shall be able to manage local time in a way that will allow them to coordinate data exchange with other members of a federation.

Federation designers will identify their time management approach as part of their implementation design. Federates shall adhere to the time management approach of the federation.

Annex A

(informative)

Rationale

A.1 Federation rules

A.1.1 Rule 1

Federations shall have an HLA FOM, documented in accordance with the HLA OMT.

The HLA is domain-independent and can be used to support federations for a wide variety of uses. The formalization of agreements for information exchange is an important element of the HLA. By formalizing the development of these agreements and requiring that the results be documented in a common format, the HLA provides the means for understanding the key elements of a federation and for assisting in the reuse of the federation, in whole or in part. Such reuse is a goal of the HLA. In addition, the FOM provides the basis for some of the data used to initialize the RTI for the federation.

A.1.2 Rule 2

In a federation, all simulation-associated object instance representation shall be in the federates, not in the RTI.

One basic idea behind the HLA is to separate federate-specific functionality from general-purpose supporting infrastructure. Federate functionality was separated from federation support services for several reasons. First, the RTI services are intended to be the basic set of broadly reusable capabilities needed to support federations across the widest range of users. These are essentially coordination and management services supporting federation operations, time coordination, data distribution, etc. Because they apply across a range of HLA applications, these services can be provided most cost effectively as services to the applications rather than as components of the applications. Separation of the RTI services has the added advantage of freeing the federates to focus on their primary objective of representing object instances to meet the needs of a user or application domain. This approach frees the developers of federates from investing time and resources in these basic common services.

A.1.3 Rule 3

During a federation execution, all exchange of FOM data among joined federates shall occur via the RTI.

The reason for providing common RTI services to federations is to provide in common the needed basic functionality to permit coherency in data exchange among the joined federates. If a federation was to exchange data representing state changes of shared object instances or interactions outside of the RTI service suite, the coherency of the distributed application would be violated.

A.1.4 Rule 4

During a federation execution, joined federates shall interact with the RTI in accordance with the HLA interface specification.

By requiring a standardized, common interface between federates and the RTI, along with a common API, the HLA allows for independent development and implementation of federate applications. Federate developers can work independently and develop interfaces to the RTI without regard to RTI implementation, and RTI developments can proceed without explicit consideration of federate development. The separation of the interfaces from the requirements for federate data exchange allows for the reuse of a common interface specification across the broad spectrum of distributed M&S applications, with specific application needs tailored through the FOM mechanism.

A.1.5 Rule 5

During a federation execution, an instance attribute shall be owned by at most one joined federate at any given time.

The HLA also provides the mechanism during federation execution to dynamically transfer ownership of an instance attribute from one joined federate to another or for a joined federate to unconditionally divest its ownership of an instance attribute. In this second case, the instance attribute becomes "unowned" by all joined federates, hence the rule explicitly states that "an instance attribute shall be owned by *at most one* joined federate at any given time." Joined federates can request ownership of "unowned" instance attributes. By defining ownership at the instance attribute level and providing the tools to hand off ownership during federation execution, the HLA provides a flexible tool set for using various combinations of joined federates to meet user needs.

A.2 Federate rules

A.2.1 Rule 6

Federates shall have an HLA SOM, documented in accordance with the HLA OMT.

A major goal of the HLA is to support interoperability and reuse of simulations. The HLA provides this support by providing for reuse at the level of simulations (or, more generally, federates) and allowing access to the representations in those simulations.

Lack of cost-effective access to information about the object representations available in federates inhibits reuse. The requirement for a SOM addresses information access by requiring federates to document the minimum essential, salient aspects of their capabilities to allow for easy identification of the federates' potential application in new federations. Although the full set of information required by a potential user will go well beyond the SOM contents, providing easy access to characterizations of simulation based on reuse potential will enable users to decide more effectively whether to invest in further assessment of the simulations to their applications. There are certain circumstances, such as dynamically configured passive subscribers, where a federate's SOM is based on the current FOM.

A.2.2 Rule 7

Federates shall be able to update and/or reflect any instance attributes and send and/or receive interactions, as specified in their SOMs.

By designing federates from the outset with the ability to present internal objects/attributes/interactions as public, the mechanisms for reuse of the federate will be in place from the start.

A.2.3 Rule 8

Federates shall be able to transfer and/or accept ownership of instance attributes dynamically during a federation execution, as specified in their SOMs.

By building in the functionality to transfer and accept ownership of instance attributes, federates designed in accordance with the HLA provide some of the basic structural tools to become federates in the widest possible range of future federations. With this capability, it is possible to allow a federate designed for one purpose to be coupled with one designed for another purpose to meet a new requirement.

A.2.4 Rule 9

Federates shall be able to vary the conditions (e.g., thresholds) under which they provide updates of instance attributes, as specified in their SOMs.

By designing and building in the capability to vary the conditions under which they can export instance attributes, federates designed in accordance with the HLA provide some of the basic structural tools to become federates in the widest possibly range of future federations.

A.2.5 Rule 10

Federates shall be able to manage local time in a way that will allow them to coordinate data exchange with other members of a federation.

The HLA time management structure is intended to support interoperability among federates that may use different internal mechanisms. Different categories of federates typically use a subset of an RTI's services to implement their mechanisms. Federates do not explicitly indicate to the RTI how time flow is being managed (time stepped, event driven, external time synchronization, independent advance, etc.) and need not use the services described in Clause 8 (*Time Management)* of IEEE Std 1516.1-2000. The HLA provides a single, unifying approach to time management, in that it offers a "toolbox" of services that support (not guarantee) federations in managing their event synchronization in a way that will meet both the needs of the federation and the range of internal time management strategies of the member federates.

Annex B

(informative)

Bibliography

[B1] IEEE 100, The Authoritative Dictionary of IEEE Standards Terms, Seventh Edition.