Overview about the High Level Architecture for Modelling and Simulation and Recent Developments

Dr. Steffen Straßburger
Fraunhofer Institute for Factory Operation and Automation
Sandtorstr. 22, 39106 Magdeburg

Abstract

The High Level Architecture for Modeling and Simulation, or HLA for short, is an IEEE standard for distributed simulation. It focuses on interoperability and reusability of the components (called federates) and offers time management interoperability as well as sophisticated data distribution concepts.

HLA has its origin in the military simulation community where one of its major tasks is the networking of military training simulators. However, due to its openness and generic character it also has a large impact on non-military distributed simulation applications. Due to these facts, HLA can still be regarded the state-of-the-art standard for distributed simulation.

This article introduces the background and history of HLA, introduces its main concepts, and discusses recent developments. A summary and evaluation of the future of HLA concludes this contribution.

1. Introduction and Motivation

The development of the High Level Architecture for Modeling and Simulation (HLA) was initiated by the U.S. Department of Defense (DoD) in 1995 out of the need for a common high-level simulation architecture. The standard was supposed to facilitate the interoperability and reusability of all types of simulation used and sponsored by the DoD.

The necessity of the standard is derived from the complexity and variety of simulation applications in use and the manifold of expectations towards simulation applications. They include different levels of abstraction, different levels of interactivity, different temporal behavior, etc. In essence, no single monolithic simulation application could fulfill all requirements of all users. Considering the different simulation applications in use, no one could foresee all their potential usage and combinations in advance. Thus, the idea of a modular, composable approach for building federations of simulations was born which eventually led to the development of the HLA.

HLA's main objective was to provide an open architecture offering services for interoperability and reusability. The architecture has no limitations towards a specific simulation paradigm. It is not even limited to simulation applications, rather it offers interoperability to all kinds of programs. However, HLA provides specific interoperability support services to accommodate specific needs of simulation applications. With that, HLA supersedes general interoperability standards like CORBA or DCOM.

Initiated as a standard in the military simulation community the development of the HLA has been overseen by the Defense Modeling and Simulation Office (DMSO) for the U.S. DoD. DMSO has deliberately taken a very open approach in the definition and accessibility of HLA and has sponsored publicly available software implementations of the HLA software.

With this policy DMSO has ensured a broad community involvement in the development of HLA, which can be seen as a cornerstone to its rather good acceptance and adoption. In the military simulation domain HLA is a mandatory standard not only in the U.S., but also throughout most NATO countries.

HLA involvement of the civilian simulation community has mostly originated from academia [1] and has been rather research oriented. Significant efforts have been focused on using HLA as a standard for interoperability between commercial of the shelf simulation packages [2,3,4].

An important joint research project in the field of civilian HLA applications was the IMS Mission project. Its focus was on adopting HLA as a standard for design, planning and operation of globally distributed enterprises. One outcome was a concept and solution for distributed supply chain simulation [5].

Serious practical applications of HLA have been investigated by several companies, among them DaimlerChrysler in the automotive sector [6]. Especially for the automotive industry with its large supplier networks and rather advanced use of digital planning and simulation methods within their Digital Factory efforts, HLA can play a substantial role for providing plug-and-play simulation interoperability.

2. A Short History of HLA

Research in the field of distributed simulation has a long tradition. Parallel distributed event simulation (PDES) is one important branch of distributed simulation primarily driven from the civilian simulation community which aims at performance and speedup issues. Conservative and optimistic synchronization protocols were developed to handle possible causality violations between simulations.

In the military simulation community, the Distributed Interactive Simulation (DIS) technology was developed primarily for the connection of real-time training simulators. DIS is defined in the IEEE 1278 standard since 1993. Another standard for the connection of constructive military simulations was the Aggregate Level Simulation Protocol (ALSP).

Outside the simulation domain, standards for distributed computing like the Parallel Virtual Machine (PVM) and the Message Passing Interface (MPI) have been developed which also influenced the field of distributed simulation.

The HLA combines its predecessor technologies from the military sector, DIS and ALSP, and is the designated standard architecture for all U.S. DoD modeling and simulation activities. The HLA development started in 1995 with the DoD Modeling & Simulation Master Plan which demanded to "establish a common high-level simulation architecture to facilitate the interoperability of all types of models and simulations ..., as well as to facilitate the reuse of M&S components".

The timeline of the development of the HLA is depicted in Figure 1. The base definition of the HLA 1.0 Standard in August 1996 can be regarded the first stable HLA definition. Shortly after its release, different versions of the RTI software developed under DMSO sponsorship became publicly available. This software was distributed freely in the community and included an RTI Help Desk as a support infrastructure for maintaining the software.

The next major release of the HLA standard was HLA 1.3 in February 1998. This version of the standard is still quite commonly in use today in many simulation applications. Two RTI developments following this 1.3 release of the standard were made publicly available in the following time, the first being RTI 1.3 in 1998, the next being RTI 1.3 NG in 1999. The latter release offered improved performance and is still available today from the Virtual Technology Corporation as RTI NG Pro. The HLA 1.3 release formed the base for further standardization efforts. Among them was the adoption of HLA by the OMG as facility for distributed simulation.

The most important standardization activity was the release of the IEEE version of the HLA standard. This release is in most parts similar to HLA 1.3, but contains several needed improvements which surfaced in the practical use of HLA 1.3 [7]. Also, some modifications needed in order to comply with IEEE requirements were made.

The year 2002 marked the end of a transition phase in which DMSO had led (and sponsored) the efforts to develop HLA. Having become an IEEE standard the further development of HLA was given into the hands of the Simulation Interoperability Standards Organization (SISO).

SISO originated over ten years ago with the Distributed Interactive Simulation (DIS) Workshops and was since that time focused on creating standards for simulation interoperability. SISO is a volunteer organization with members from industry, military and academia.

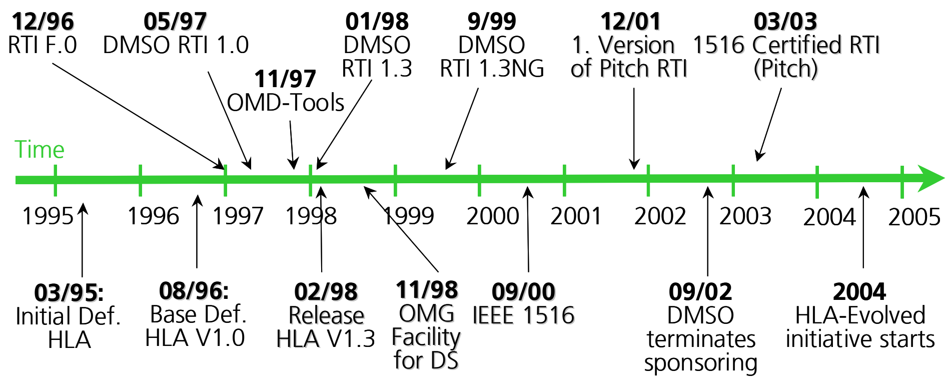

Figure 1: HLA Development Timeline

Besides hosting the Simulation Interoperability Workshops (SIW) which are organized three time a year (two in the U.S.A, one in Europe) SISO hosts a Standards Activity Committee (SAC) which oversees the work of several Product Development Groups (PDG). PDGs are the actual groups of people developing standards for simulation interoperability. Most of their work is based on HLA, including its future refinement and development of standards based on top of HLA.

The most important PDG is the HLA-Evolved initiative as it oversees the review of the IEEE 1516 specification. Many new potential HLA requirements have been identified based on feedback from the various domains and application areas. The PDG seeks to address these requirements via a formal open review of the IEEE 1516 series of specifications. As part of this process, the PDG will incorporate those aspects raised in the DoD Interpretations Document for IEEE 1516 [8] and a Dynamic Link Compatible HLA API for IEEE 1516.1.

3. Major Concepts of HLA

In order to facilitate interoperability and reusability HLA differentiates between the simulation functionality provided by the members of the distributed simulation and a set of basic services for data exchange, communication and synchronization. Figure 2 gives an functional overview of a distributed simulation under the HLA paradigm.

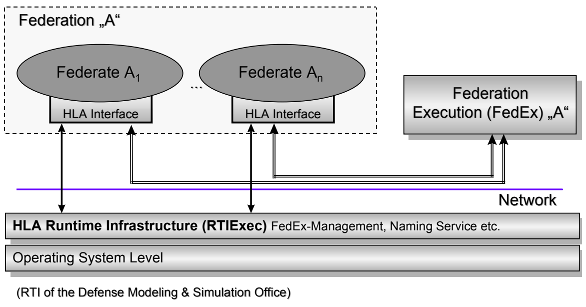
Federation „A“
Federate A_1
Federate A_n
HLA Interface
HLA Interface
Federation Execution (FedEx) „A“
Network
HLA Runtime Infrastructure (RTIExec) FedEx-Management, Naming Service etc.
Operating System Level
(RTI of the Defense Modeling & Simulation Office)

Figure 2: Functional view of a distributed simulation under HLA

In HLA, individual simulations and other participants of a distributed simulation are referred to as federates. Federates which are supposed to cooperate together under certain guidelines and a defined object model form a so-called federation. Federates use a common runtime infrastructure (RTI) for communication. The RTI is a piece of software which can be regarded as a distributed operating system add-on. HLA defines a bi-directional interface between federates and the RTI. A single run of the federation is referred to as a federation execution.

The current version of the High Level Architecture for Modeling and Simulation, or HLA for short, is formally defined in the three key documents of the IEEE standard 1516. These documents are

- 1516-2000: Framework and Rules
- 1516.1-2000: Federate Interface Specification
- 1516.2-2000: Object Model Template (OMT) Specification

All three elements are briefly discussed in detail in the following sections.

3.1 HLA Rules

The HLA Rules define the required behavior of a federation and its federates and are thus part of the formal HLA compliance definition. There are 5 rules for federations and 5 for federates, which are stated below [11].

Rules for federations:

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 runtime infrastructure (RTI).
3. During a federation execution, all exchange of FOM data among 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.

Rules for federates:

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.

3.2 HLA Federate Interface Specification

The HLA Federate Interface Specification describes the services which federates have to use for communicating with other federates via a runtime infrastructure (RTI). The interface specification describes which services can be used by a federate and which services it has to provide [12].

This bi-directional character of the interface is encapsulated into an ambassador paradigm. A federate communicates with the RTI using its RTI ambassador. Conversely, the RTI communicates with a federate via its federate ambassador. From the federate programmer's point of view, these ambassadors are objects and the communication among the participants is performed by calling methods of these objects. Thus, the services defined in the interface specification are either methods of the RTI ambassador or of the federate ambassador.

The interface specification defines six categorizes of services, which will be briefly described in the following sections. A special advantage of HLA compared to other technologies are the time management and the data distribution management services.

The time management services provide a mechanism for coordinating simulation clocks of simulations using a wide variety of time advance mechanisms. In comparison with other technologies, where time management/synchronization is only available to a certain type of simulation, HLA provides a general solution for all types of simulations.

The services provided in the data distribution category provide new mechanisms for efficiently transferring data among certain federates and for reducing the amount of data transferred. They are special in that regard, in that previous technologies (like DIS) are usually based on broadcast principles for distributing data.

3.2.1 Federation Management

The main focus of the services in the Federation Management service group is the coordination of federation-wide activities during a federation execution. They are used by federates to initiate, join, resign, and manage a federation execution.

Interface services include:

- Create/Destroy Federation Execution: These service are used to create and destroy federation execution. Usually the first federate joining a federation execution has the task of creating it. The last federate leaving a federation execution commonly destroys it.
- Join/Resign Federation Execution: These services are used by federates to join a federation execution and to resign from it once the federate has completed its tasks.
- Services to save and restore federation executions: These services can be used to save and restore the state of the federation. It should be noted that these service only coordinate the save/restore process. The internal state saving mechanisms have to be implemented by the federates themselves.

3.2.2 Declaration Management

Federates shall use declaration management services to declare their intention to generate and receive information. A federate shall invoke appropriate declaration management services before it can register or discover object instances, update or reflect instance attribute values, and send or receive interactions.

With that, declaration management could also be seen as an “interest management“. Federates specify, which data types they would like to send or receive. The publishing and subscribing of data types (object and interaction classes with their attributes and parameters) has to be performed in accordance with the SOMs and the FOM. Although declarations can be changed dynamically during a federation execution, the declaration management belongs to the initialization phase of a federation.

Interface services include:

- Publish Object Class Attributes/Interaction Class: These services are used to announce that a federate intends to generate the specified object and interaction classes “later on“ during a federation execution.
- Subscribe Object Class Attributes/Interaction Class: These services are used to announce that a federate is interested in the specified object and interaction classes and would like to receive information about these classes from now on.
- Start (Stop) Registration For Object Class/Turn Interactions On (Off): Using these callback functions to the federate, the RTI can inform a federate whether other federates are interested in the object classes and interactions it has published. These services implement the so-called “advisory switches” which inform federates of the relevance of their publications. Federates can chose to ignore these switches and register object instances/ send interactions regardless of whether other federates are interested.

Figure 3 gives an example of the services that two federates might use to manage their subscription and publications.

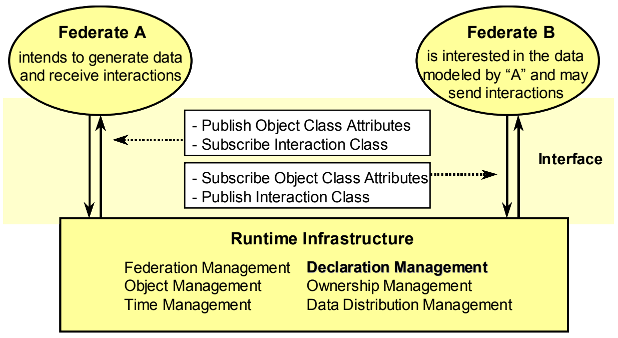
Federate A
intends to generate data and receive interactions
Federate B
is interested in the data modeled by "A" and may send interactions
- Publish Object Class Attributes
- Subscribe Interaction Class
- Subscribe Object Class Attributes
- Publish Interaction Class
Interface
Runtime Infrastructure
Federation Management
Object Management
Time Management
Declaration Management
Ownership Management
Data Distribution Management

Figure 3: Declaration management (adopted from [9])

3.2.3 Object Management

This group of the interface specification provides services for the registration, modification, and deletion of object instances and the sending and receiving of interactions. The services of this group provide the necessary functionality for all data exchange among federates.

RTI services include:

- Register Object Instance/Discover Object Instance: Each object that is relevant to a federation execution needs to be registered with the RTI using the Register Object Instance service. Interested federates will be notified of the existence of such an object instance via the Discover Object Instance callback to their federate ambassador.
- Update/Reflect Attribute Values: After informing the RTI about the existence of an object instance, the registering federate can start sending updates for this object via the Update Attribute Values service. Interested federates will receive updates via the Reflect Attribute Values callback to their federate ambassador.
- Send/Receive Interaction: Interactions can be sent via the Send Interaction service and are received via the Receive Interaction callback service.
- Delete Object Instance: This service removes an object instance from a federation execution.
- Change Transport and Ordering Mechanisms: Object updates and interactions are transported using certain transportation and ordering mechanisms which can be changed at runtime. Transportation types include reliable and best-effort transmission, ordering mechanisms include time stamp order and receive order.

Figure 4 gives an example for the usage of the services introduced in this section.

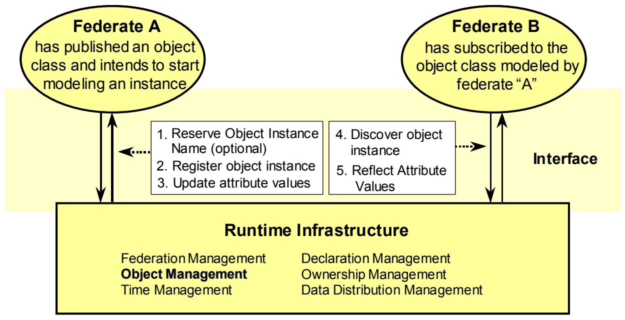
Federate A
has published an object class and intends to start modeling an instance
Federate B
has subscribed to the object class modeled by federate "A"
1. Reserve Object Instance Name (optional)
2. Register object instance
3. Update attribute values
4. Discover object instance
5. Reflect Attribute Values
Interface
Runtime Infrastructure
Federation Management
Object Management
Time Management
Declaration Management
Ownership Management
Data Distribution Management

Figure 4: Object management (adopted from [9])

3.2.4 Ownership Management

Ownership management can be used by federates and the RTI to transfer ownership of attribute instances among federates. The ability to transfer ownership is intended to support the cooperative modeling of a given object instance across a federation.

The services provided by this group support both push and pull mechanisms for ownership transfer.

RTI services include:

- Negotiated Attribute Ownership Divestiture/ Request Attribute Ownership Assumption: The service Negotiated Attribute Ownership Divestiture is intended for federates that want to wishes to divest itself of the instance attribute (push). Request Attribute Ownership Assumption is the corresponding callback to the federate ambassador for informing the federate about the request.
- Attribute Ownership Acquisition/ Request Attribute Ownership Release: The service Attribute Ownership Acquisition is intended for federates that want to become owner of a certain set of attributes (pull). Request Attribute Ownership Release is the corresponding callback to the federate ambassador for informing a remote federate about the request.
- Attribute Ownership Divestiture Notification/Acquisition Notification: These services inform the federates about the success of their respective requests.

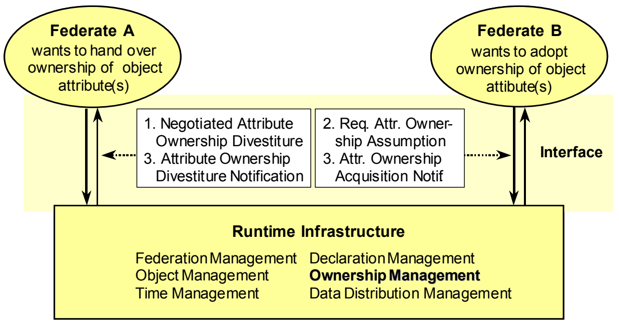
Federate A
wants to hand over ownership of object attribute(s)
Federate B
wants to adopt ownership of object attribute(s)
1. Negotiated Attribute Ownership Divestiture
3. Attribute Ownership Divestiture Notification
2. Req. Attr. Ownership Assumption
3. Attr. Ownership Acquisition Notif
Interface
Runtime Infrastructure
Federation Management
Object Management
Time Management
Declaration Management
Ownership Management
Data Distribution Management

Figure 5: Ownership management (adopted from [9])

Figure 5 illustrates the push mechanism outlined above.

3.2.5 Time Management

Time management is concerned with the mechanisms used by simulations to advance through simulation time. Time advances are coordinated by the RTI with object management services so that information is delivered to federates in a causally correct and ordered fashion.

HLA Time Management provides mechanisms to support all major types of regimes to advance simulation time, such as (scaled) real-time and as-fast-as-possible simulations.

An important design principle that is used to allow this functionality is time management transparency. This means the local time management mechanism used in a certain federate does not have to concern other federates. For instance, a federate using an event-oriented mechanism does not need to know whether the federate with which it is interacting is also using an event-oriented mechanisms, or (say) a time-stepped mechanism.

An HLA federation may even include federates using HLA Time Management services to co-ordinate their time advances, and others, that do not. In such an environment, the RTI must determine those federates that must be considered when coordinating time advances. Therefore HLA introduces two boolean flags that determine the federate’s time management characteristics. They are called time-constrained and time-regulating flags. A time regulating federate is one that wishes send time-stamped messages to other federates and thus influence their time advancement. A time-constrained federate is one that wishes to be able to receive time-stamped messages, and thus subordinates itself to the federation time advancement.

The HLA time management services are strongly related to the services for exchanging messages, e.g., attribute updates and interactions. There are two general ordering types for messages under HLA: receive-order (RO) and time-stamp-order (TSO). Receive-ordered messages are simply placed in a queue when they arrive, and are immediately eligible for delivery to the federate. TSO messages are assigned a time-stamp by the sending federate, and are delivered to each receiving federate in the order of non-decreasing time stamps. Incoming TSO messages are placed into a queue within the RTI, but are not eligible for delivery to the federate until the RTI can guarantee that there will be no TSO messages for that federate with a smaller time stamp.

In order to allow the RTI to perform time management, a federate must use one of the following time management services (as appropriate for the internal time advance mechanism of the federate).

- Next Event Request (NER)

Event driven federates need to process local and external events, i.e., events generated by other federates, in time-stamp-order. The federate time, i.e., its logical simulation time, typically advances to the time stamp of each event as it is processed. An event driven federate will typically use the Next Event Request service when it has completed all simulation activity at its current logical time in order to advance to the time stamp of its next local event.

- Time Advance Request (TAR)

Time step driven federates make time advances in time steps with some fixed duration of simulation time. The simulator does not advance to the next time step until all simulation activities within the current time step have been completed. This type of federate will usually use the Time Advance Request service to request to advance its logical time to the next time step.

- Flush Queue Request (FQR)

FQR can be used for optimistically synchronized federates to request the out-of-order delivery of events. HLA supports optimistic federates while maintaining time management transparency. Specifically, the HLA time management services do not require all federates to support a rollback and recovery capability even if one federate is using optimistic event processing. FQR is used by optimistic federates to receive all buffered messages (although there might be some messages at a later point in time which carry a smaller time stamp). In such a case the federate will have to use the other important service Retract, which can be used to cancel a previously sent message. The RTI ensures that “optimistic” messages are only received by optimistic federates, as long as there is a possibility of a later cancellation of that message.

Using one of these services a typical synchronization loop of an HLA federate would work in the following three step order:

1. Request advancement of logical time by calling the appropriate RTI service (e.g., Next Event Request, Time Advance Request)
2. Receive zero or more messages from the RTI (e.g., receive the Reflect Attribute Value or Receive Interaction callback from the RTI)
3. Receive a Time Advance Grant callback from the RTI to indicate that the federate’s logical time has been advanced.

A more detailed discussions of HLA Time Management can be found in [10].

3.2.6 Data Distribution Management

The data distribution management (DDM) services provide a mechanism to reduce both the transmission and the reception of irrelevant data. Whereas declaration management services provide information on data relevance at the class attribute level, data distribution management services add a capability to further refine the data requirements at the instance attribute level.

This is achieved be defining multi-dimensional routing spaces. The producers of data (the sending federates) are expected to specify an update region associated with a specific attribute update or interaction. This is the region in which the update or interaction is relevant. Receiving federates have to specify which regions they are interested in (subscription regions). The actual data transfer only takes place if the update and subscription regions for a specific update or interaction overlap (Figure 6).

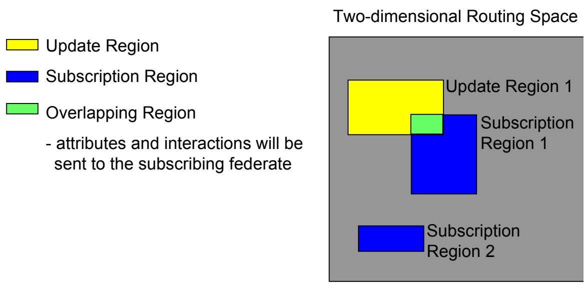

Figure 6: Data distribution management – example of routing spaces

The usage of DDM is optional, but provides a sophisticated means for the minimization of the amount of transferred data and thus the network load.

3.3 HLA Object Model Template Specification

The Object Model Template (OMT) defines the way in which federations and federates have to be documented. The HLA object models are the formal definition of the data that is transferred between federates [13] and thus are one of the main vehicles for interoperability in HLA.

While the HLA interface specification provides for the technical interoperability between software systems regardless of platform and language (the „transmission line“), the object model template (OMT) defines the “language” spoken over that line.

HLA applies an object oriented world view which is slightly different from the one known from the area of object oriented programming (OOP). In HLA, two types of classes exist: object classes and interaction classes. Object classes describe the simulated entities with their attributes. Interaction classes describe the relationships between different object classes, i.e., their interactions, and can have parameters associated with them. In contrast to OOP, HLA object models do not specify the methods of objects, since in the common case the behavioral description is nothing that needs to be transferred between federates.

It should be noted that this object oriented world view does only define how federates have to represent themselves to other federates. The object oriented world view does not dictate any internal representation inside the federate, i.e., it merely defines the interface to the outside world.

The HLA specification requires that each individual federate provides a so-called simulation object model (SOM) which is produced according to the OMT. The SOM of a federate defines its modeling capabilities in terms of what kind of data the federate is providing to other federates and what it is expecting to receive from others.

In addition to each federate’s SOM, the HLA specification also requires that for each federation a so-called federation object model (FOM) is provided. The FOM is a superset of the information from the individual SOMs of the federates. It thus contains all the classes defined by the individual participants of the federation and gives a description of all shared information. The FOM can be seen as a contract among “n” simulations to satisfy the objectives of a specific federation.

In general, the object models under HLA describe:

- The set of objects chosen to represent the real world for a specific simulation/federation
- The attributes, associations, and interactions of these objects
- The level of detail at which these objects represent the real world, including spatial and temporal resolution

Both the SOM and the FOM are based on a format specified in the OMT, which is a general template specifying the tables that need to be documented. The following list explains the most important ones:

- Object Class Structure Table: This table lists the namespace of all simulation/federation object classes and describes their class-subclass relationships. Thus it contains the (static) object class descriptions of a federate/federation and supports hierarchical class structures. There is no mechanism for multiple inheritance.
- Interaction Class Structure Table: This table describes the “dynamics” among objects by depicting all possible types of interactions among them. It also supports class-subclass relationships.
- Attribute/Parameter Table: This table gives detailed information about objects and interactions by specifying the “features“ of object attributes and interaction parameters in a simulation/federation.
- Data Type Table: This table specifies details of the data representation in the object model. This is esp. important since HLA allows the specification of complex and enumerated datatypes which must be documented here.
- FOM/SOM Lexicon: Each term listed in one of the above tables (e.g., object class names) has to be described in a verbal form in this part of the OMT. The FOM/SOM lexicon is essential to ensure that the semantics of the terms used in an HLA object model are understood and documented.

The issue of defining semantic interoperability is very difficult to solve. For a general, non-application specific architecture like HLA, it is necessary to keep application specific definitions separated from the actual architecture definition. In its predecessor DIS this guideline had not been taken account of, leading to a mixture of network protocol and application specific definitions.

In HLA a strict separation of syntactic and semantic interoperability has been followed. HLA provides the syntax for interoperability. For solving the semantic interoperability, HLA provides the framework for its definition, i.e., the templates for establishing the object models (SOM and FOM), but the task of filling the contents is left to the federation developers.

Since establishing FOMs and agreeing upon common definitions and understandings of certain terms which might be contained in a specific FOM is an effort and time consuming task, the notion of reference FOMs was soon introduced. Reference FOMs are not part of the actual HLA definition. They are usually established by a group of people from a certain niche of the simulation community, summarizing all the semantic definitions agreed upon in this group. One example for such a reference FOM is the Real-time Platform Reference FOM (RPR-FOM), which has been developed in one of the SISO PDGs. The RPR-FOM provides the definitions commonly used in the real-time simulation community.

The process of developing object models is supported by different existing tools (e.g., the Object Model Development Tool (OMDT) by AEgis, the Visual OMT by Pitch). These tools provide an intuitive user interface for creating object models and allow the conversion between the HLA 1.3 format of the OMT and the new XML-based IEEE 1516 representation.

4. Recent Developments

This section introduces recent developments and ongoing efforts that go beyond the existing HLA standard trying to improve it or to come up with alternative solutions.

4.1 COTS Simulation Package Interoperability

Soon after the creation of HLA it became obvious that its applicability would not be limited to military applications. A majority of HLA's concepts could also form the basis for a much needed simulation interoperability standard in the civilian simulation community, the manufacturing, logistics, and transportation industry being example target application areas.

Since simulation models in industry are mainly designed and developed in commercial-off-the-shelf (COTS) simulation packages, the prerequisites in this sector are different. As HLA itself is not focused on coupling models created in COTS simulation packages, ways have been investigated to adopt HLA for the usage with these packages [2,3,14]. Principal solutions have been developed for several packages, SLX being one of the first [15].

Ongoing standardization activities concentrate of defining *standardized ways* of providing HLA based interoperability for COTS packages. The challenge here is not in adopting HLA as such, but coming up with an easy and standardized approach for a certain class of simulation problems. The necessity of these efforts is derived from the different possibilities of using the HLA standard, e.g., a simple matter like entity passing from one model to another can be solved in different ways: an entity being passed could be modeled as an HLA interaction sent from a sink in the first model to a source in the second model. An equivalent solution could model the entities as HLA object instances and use ownership management services to pass the entities [16]. Both solutions are valid HLA-based solutions, but they are not interoperable.

The SISO PDG on Commercial Off-the-Shelf Simulation Package Interoperability is devoting its efforts to solving these problems [17]. Their approach is based on establishing so-called interoperability reference models (IRM). These IRMs describe different classes of commonly faced problems when adopting HLA for a COTS package and a standardized way to solve them. It is anticipated that COTS package vendors adopt these IRMs when creating HLA interfaces for their packages, thus achieving full interoperability for the designated problem classes.

4.2 SISO Standardization Activities

SISO is devoted on continuously creating standards and solutions for simulation interoperability issues. As already discussed, the HLA-Evolved PDG is in charge of revising the HLA standard within the cyclic 5-year review process of IEEE. Other important PDGs are briefly described in the following.

- BOM PDG - Base Object Model Specification

Base Object Models (BOMs) can be considered as reusable packages of information representing independent patterns of simulation interplay, and are intended to be used as building blocks in the development and extension of simulations [18].

The BOM is intended as a component-based standard for describing a reusable piece part of a federation or an individual federate. BOMs provide developers and users a modular approach for defining and adding new capabilities to a federate or federation, and in quickly composing object models. BOM elements include object classes, interaction classes, patterns of interplay, state machines, and events.

- SRML PDG - Simulation Reference Markup Language

SRML is an XML-based language for describing and executing web-based simulation models. SRML is defined as an XML schema that adds simulation behavior to XML documents.

The intention of the PDG is to standardize SRML as the interchange of self-describing simulation models that include content and behavior, as well as standardizing the description of how that language would operate in a simulator. The fundamental premise is that an open XML-based standard language and execution environment for simulations will benefit the simulation industry in a similar way to that in which the standardization of HTML and the web browser have benefited the general computing industry.

4.3 Standardization Activities outside SISO

The German Armed Forces Technical Centre for Communications and Electronics, WTD 81, has taken a very active role in the evaluation of the applicability of HLA and has introduced several interesting concepts that go beyond it.

Initially, the efforts started with the sponsored development of GERTICO. GERTICO is an acronym for “GErman Run-Time Infrastructure based on COrba”. With this effort, the WTD has propagated its preference for building HLA on top of an existing well-known standard like CORBA.

In a related effort, the WTD has devised the concept of pSISA. pSISA is the Proposed Standard Interface for Simulation Applications. Its purpose is to foster the reusability of the HLA interface code and to ease the implementation of HLA compliant applications. The major design goal is the complete encapsulation of the RTI in a object oriented shell. The application programming interface (API) accessible to the application is largely based on the object model to convey. As a result, the API is object oriented and can be built by a code generator from the HLA simulation object model (SOM) to provide C++ classes in one-to-one correspondence with the SOM object and interaction classes.

Simulations built on pSISA are thus expected to be independent from the underlying communication infrastructure. The latter can be based on CORBA, HLA, or any other upcoming standard [19].

Further discussions and standardization efforts outside SISO are driven by several national groups outside the USA, the HLA Competence Centre in Magdeburg, Germany with its annual HLA-Forum being one of them [20].

5. Evaluation and Summary

This section attempts to give an unbiased evaluation of the current state of HLA and its potential for the future.

5.1 Is HLA worth the effort?

An often heard opinion about HLA is that it is too complex to use, too heavy in its performance characteristics and that there is too much overhead involved in using it.

Admittedly, HLA with its federate interface specification is indeed one of the most complex standards out there. Consequently, there is certainly a rather high learning curve between getting a first glance at the standard and having the first federation running.

However, with the objectives in mind that HLA was developed for, the author has the strong conviction that it would be difficult if not impossible to devise a standard with less complexity still fulfilling all requirements HLA fulfills. The answer to the question, if it is always HLA that is needed to network two simulators highly depends on the circumstances. If there is a very high certainty that the two simulators must be networked for a single purpose only and it is highly unlikely that they ever be re-used again, then there are certainly solutions for networking them with less effort and overhead. Otherwise the effort for creating a standardized interoperable version of both simulators with a high degree of reusability is certainly worth it.

5.2 Will HLA ever become a mainstream technology?

Having been involved with HLA on both sides, academia and industry, the author postulates the thesis that HLA is far from becoming a mainstream standard for civilian simulation applications. This also concurs with the observation made by Boer [14] in his PhD thesis. One main reason can certainly be seen in the degree of simulation usage itself. Even though much progress has been made, simulation is still a niche technology not applied in day-to-day business in today’s industry.

Take the example of the automotive industry: Simulation has received a major push with the digital factory initiatives of many OEMs. Still, their efforts are mainly focused on the introduction of digital planning methods. Digital planning also involves more simulation than in earlier times, but it is still tackled with a monolithic approach: All data is centrally stored in one planning database. Planning and simulation is bound to the tools of one software vendor. Interoperability between vendors and their (simulation) tools is not yet an issue which is on the demand list of the OEMs. Considering the increasing globalization and networking with supplier structures, it will become an issue rather soon.

Therefore the author has the strong believe that there is a rather good potential for usage of HLA in civilian applications. However, it will only become a mainstream technology if the standard is incorporated into commercial simulation systems by its vendors. This is the prerequisite for making it a commodity technology that can be used like any other plug-and-play standard today.

5.3 Summary

Looking back at about 10 years of history, HLA can certainly be regarded a success. It continues to be the leading simulation interoperability standard and is constantly being maintained and improved by the community itself.

HLA's open approach to define interfaces and functionalities of an infrastructure software rather than providing a black box implementation has allowed software vendors to create their own HLA implementations and become established in the distributed simulation market.

Although HLA adoption in the non-military sector has been rather cautious, good work is underway to standardize user-friendly HLA-usage with COTS simulation packages. The adoption of the HLA standard by COTS package vendors will be the prerequisite for continuing HLA's success in this community.

References

[1] Schulze, T., S. Straßburger, U. Klein: Migration of HLA into Civil Domains: Solutions and Prototypes for Transportation Applications. In: SIMULATION, Vol. 73, No. 5, pp. 296-303, November 1999.

[2] Straßburger, S.: Distributed Simulation Based on the High Level Architecture in Civilian Application Domains. Ghent: SCS-Europe BVBA, 2001. ISBN 1-565552180.

[3] Ryde, M.D. and Taylor, S.J.E.: Issues in Using COTS Simulation Packages for the Interoperation of Models. In: Proceedings of the 2003 Winter Simulation Conference, eds. S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, pp. 772-777. December 7-10, 2003. New Orleans, USA.

[4] Taylor, S., B. Gan, S. Straßburger, A. Verbraeck: HLA-CSPIF Panel on Commercial Off-the-Shelf Distributed Simulation. In: Proceedings of the 2003 Winter Simulation Conference, eds. S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, pp. 881-887. December 7-10, 2003. New Orleans, USA.

[5] Rabe, M., F.-W. Jaekel: Non Military use of HLA within Distributed Manufacturing Scenarios. In: Proceedings Simulation und Visualisierung '01, (Eds.) T. Schulze, V. Hinz, S. Schlechtweg. Magdeburg, 22.03-23.03. 2001, SCS International, pp. 141-150.

[6] Straßburger, S., G. Schmidgall, S. Haasis: Distributed Manufacturing Simulation as an Enabling Technology for the Digital Factory. In: Journal of Advanced Manufacturing Systems (JAMS). Vol. 2, No. 1 (2003) 111-126.

[7] Pitch Technologies AB: Differences between HLA 1.3 and HLA 1516. Available online at http://www.pitch.se/hla/abouthla1516.asp

[8] DoD Interpretations of the IEEE 1516-2000 series of standards, IEEE Std 1516-2000, IEEE Std 1516.1-2000, and IEEE Std 1516.2-2000. Available online at https://www.dmso.mil/public/library/projects/hla/rti/DoD_interps_1516_Release_2.doc

[9] Dahmann, J.: HLA Tutorial. 1997 Spring Simulation Interoperability Workshop, Mar. 3-7, 1997, Orlando.

[10] Fujimoto, R. M.: Parallel and Distributed Simulation Systems, Wiley Interscience, 2000.

[11] IEEE 1516-2000: IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules.

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

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

[14] Boer, C.A.: Distributed Simulation in Industry. Rotterdam: Erasmus Research Institute of Management, 2005. ISBN 90–5892–093–3.

[15] Straßburger, S., T. Schulze, U. Klein and J.O. Henriksen: Internet-based Simulation using off-the-shelf Simulation Tools and HLA. In: Proceedings of the 1998 Winter Simulation Conference, eds. D.J. Medeiros and E. Watson, pp. 1669-1676. SCS, Washington, D.C.

[16] Straßburger, S., A. Hamm, G. Schmidgall, S. Haasis: Using HLA Ownership Management in Distributed Material Flow Simulations. In: Proceedings of the 2002 European Simulation Interoperability Workshop. June 2002. London, UK.

[17] SISO-Homepage of the CSPI-PDG. Available online at http://www.sisostds.org/index.php?tg=articles&idx=More&article=43&topics=21

[18] BOMs Homepage. Available online at http://www.boms.info/

[19] Usländer, T., R. Herzog, K. Pixius, H.-P. Menzler: A CORBA infrastructure plugged into a German pSISA architecture. Simulation Interoperability Workshop, Fall 2000.

[20] HLA-Kompetenzzentrum Magdeburg. Available online at http://www.kompetenzzentrum-hla.de/