Judith S. Dahmann, James O. Calvin, and
Richard M. Weatherly

Flexible, interacting simulations provide a foundation for successful DMT applications.

A REUSABLE ARCHITECTURE for SIMULATIONS

The High Level Architecture (HLA) for simulation systems is designed to facilitate reuse and interoperability of a wide range of virtual and live systems in Distributed Mission Training. HLA is based on the premise that no single simulation can satisfy the requirements of all uses and users. An individual simulation or set of simulations developed for a given purpose can be applied to another application under the HLA concept of federation: a composable set of interacting simulations. The intent of HLA is to provide a structure that will support reuse of capabilities available in different simulations.

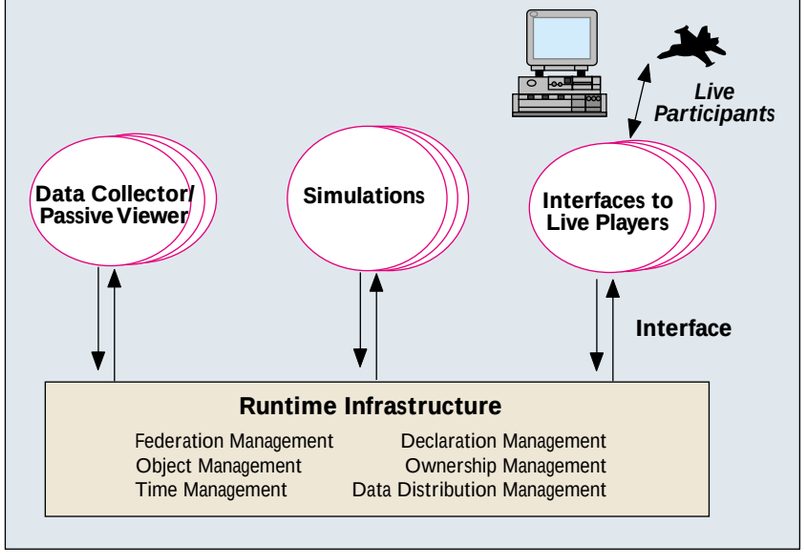
HLA has wide applicability across a full range of simulation areas, including education, training, analysis, engineering, and even entertainment. These widely differing applications indicate the variety of requirements considered in the development and evolution of HLA. DMT is one among the many applications that are based on HLA.

HLA does not prescribe a specific implementation, nor does it mandate the use of any particular software or programming language. Over time, as technology advances, new and different implementations will be possible within the framework of HLA. The motivation behind HLA is a common architecture will allow leveraging investments in simulation capabilities to meet new and changing user needs. Further, by standardizing only key elements of the architecture and not implementation, supporting software developments can be tailored to the performance needs of applications. These can also capitalize on the emerging improvements in processor and network technologies. This article outlines the key features of HLA, the rationale for its design and development, and its application to DMT. In particular, the discussion focuses on issues facing the DMT community and the ways that HLA supports the development of approaches to them.

The HLA development process involved gov-

ROBERT NEUBECKER

Figure 1. Functional view of an HLA federation.



ernment, academia, and industry. In 1995, three industry teams developed concepts for the definition of a high-level architecture. The results of these efforts were combined along with additional insights from other modeling and simulation projects to arrive at the initial definition of HLA. This definition was presented to an Architecture Management Group (AMG) formed to develop the HLA from this initial definition. AMG developed the architecture based on cooperative efforts to apply HLA in a series of prototypes designed to ensure it addresses the broadest set of application requirements. The result was a baseline HLA definition in 1996. Subsequently, AMG has continued to evolve HLA based on experience with its use, reviewing and updating the HLA specifications on a six-month cycle. In 1998, version 1.3 of the HLA specification was adopted [1]. These specifications form the basis for a draft IEEE standard for simulation interoperability architecture [3]. HLA is already a standard for use in the U.S. Department of Defense [6], has been accepted for standardization in NATO [4], and has been adopted by the Object Management Group (OMG) as the OMG Distributed Simulation Facility [5].

Figure 1 shows how an HLA federation is partitioned into its major functional components. The first key component is the simulations themselves, or more generally, the *federates*. A federate can be a computer simulation, a manned virtual training platform, a supporting utility (such as a viewer or data collector), or even an interface to a live player or instrumented facility. In HLA, all object representation is in the federates. It imposes no constraints on what is represented in the federates or how it is represented. However, HLA does require all federates incorporate specified capabilities allowing objects in the simulation to interact with objects in other simulations through the exchange of data supported by services implemented in the Runtime Infrastructure (RTI).

The second functional component is the RTI—a distributed operating system for the federates. It provides a set of general purpose services that support federate-to-federate interactions. All interactions among the federates, go through the RTI.

The third component is the interface to the RTI. The HLA runtime interface specification defines the services to be provided by the RTI and the federates, under HLA. It provides a standard way for federates to interact with the RTI, to invoke the RTI services to support runtime interactions among federates and to respond to requests from the RTI. This interface is independent of the implementation of the RTI software and is independent of the specific object models and data exchange requirements of any federation.

HLA Definitions

HLA is formally defined by three components: the interface specification, the object model template, and HLA rules.

The HLA Interface Specification describes the runtime services provided to the federates by the RTI, and by the federates to the RTI. There are six classes of services. Federation management services offer basic functions required to create and operate a federation. Declaration management services create the relationship between producers and consumers of information within an HLA federation by defining what data a federate intends to produce and what data it expects to consume. Object management services provide creation, deletion, identification, and other services at the object level. Ownership management services support the dynamic transfer of ownership of object attributes during an execution. Time management services support synchronization of simulation data exchanges. Finally, data distribution management services support the efficient routing of data among federates during the course of a federation execution. The HLA interface specification defines the way these services are accessed, both functionally and in an application programming interface (API). At present, APIs in CORBA IDL, C++, Ada 95, and Java are incorporated in the interface specification.

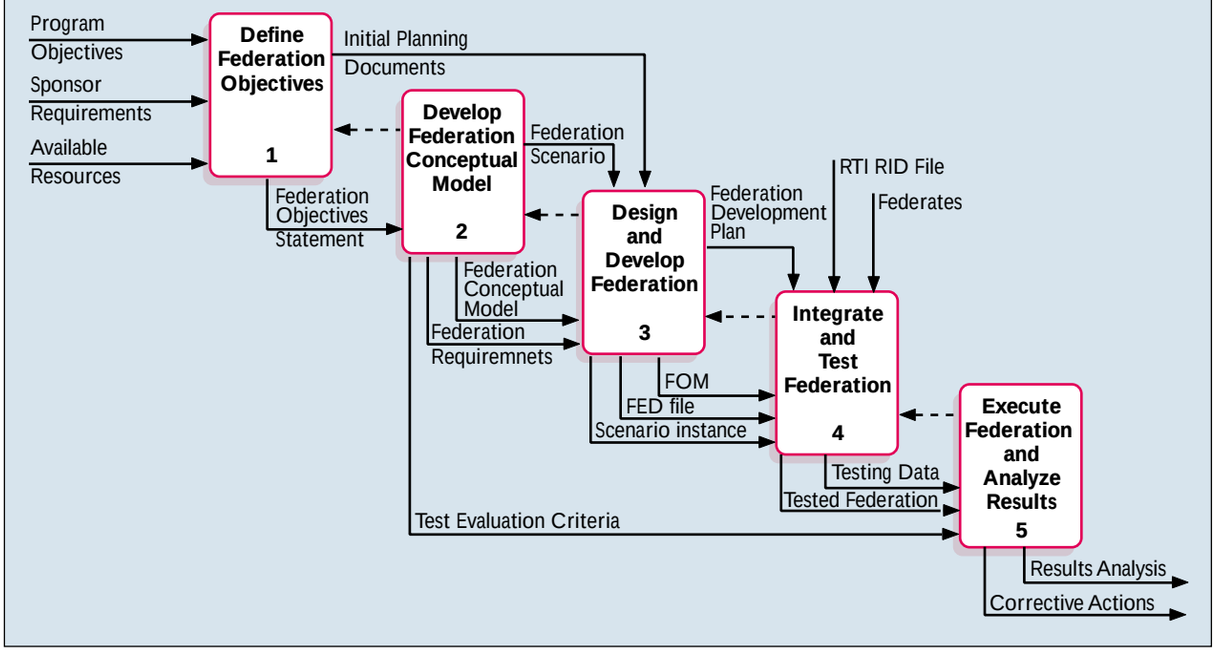
HLA Object Models are descriptions of the essential sharable elements of the simulation or federation expressed in "object" terms. HLA is directed toward interoperability; hence, object models are intended to focus on the description of data shared across a federation. HLA introduces no constraints on the content of the object models. However, HLA does require each federate and federation document its object model using a standard object model template. These templates are intended to be the means for open information sharing across the community to facilitate reuse of simulations. These completed templates will be openly available, and tools are being developed to allow for automated search and reasoning about object model template data, to further facilitate cost-effective information exchange and reuse.

HLA specifies two types of object models: the HLA Federation Object Model (FOM) and the HLA Simulation Object Model (SOM). The HLA FOM describes the set of objects, attributes and interactions, which are shared across a federation. The HLA SOM describes the simulation (federate) in terms of the types of objects, attributes, and interactions it can offer to future federations. The SOM is distinct from internal design information; rather it provides information on the capabilities of a simulation to exchange information as part of a federation. The SOM is essentially a contract by the simulation defining the types of information it can make available in future federations. The availability of the SOM facilitates the assessment of the appropriateness of the federate for participation in a federation. While the HLA does not define the contents of a SOM or FOM, it does require that a common documentation approach be used. Both the HLA FOM and SOM are documented using a standard form called the HLA Object Model Template (OMT).

Finally, the HLA Rules summarize the key principles behind HLA. The rules are divided into two groups: federation and federate rules. Federations, or sets of interacting simulations or federates, are required to have a FOM in the OMT format. During runtime, all object representation takes place in the federates (not the RTI) with only one federate owning any given attribute of an instance of an object at any given time. Information exchange among the federates takes place via the RTI using the HLA interface specification. Additional rules apply to individual federates. Under HLA, each federate must document its public information in its SOM using the OMT. Based on the information included in their SOM, federates must import and export information, transfer object attribute ownership, updates attributes, and utilize the time management services of the RTI when managing logical time.

While HLA is a runtime architecture for simulation interoperability, attention is increasingly devoted to developing views on the processes that surround the use of HLA. In particular, the HLA federation development and execution process (FEDEP) has

Figure 2. HLA federation development and execution process.



The motivation behind HLA is that a common architecture **ALLOWS LEVERAGING INVESTMENTS IN SIMULATION CAPABILITIES TO MEET CHANGING USER NEEDS.**

been developed and is evolving based on user experience with the application of HLA. In this view, the process covers five basic steps given in Figure 2. Essentially, the FEDEP presents a systems engineering approach to the reuse of simulations using HLA as the technical foundation. The FEDEP is updated every six months based on growing user experience.

FEDEP has been used as a framework for identifying opportunities to apply automation to the process of developing and executing HLA federations. Using this framework, types of tools and common data interchange formats have been defined and are being used across the HLA community [2]. Tools have been developed that support the different stages of the FEDEP process by both government and industry.

HLA as Applied to DMT

HLA provides a software architecture for designing and fielding DMT applications. While HLA was not designed specifically for DMT, features of HLA address the needs of this community. Current and future DMT initiatives will integrate aircraft simulators developed for specific air platforms into interactive simulator networks to support mission training of geographically distributed units. A critical requirement for DMT is the ability to support changing training requirements of the users with existing training assets that are geographically distributed. Added elements will be brought into these environments depending on the particular training objectives and requirements of the training audience. These include software representations of enemy air defenses, ground units, strategic targets along with other aircraft, both enemy and friendly. Interfaces to other live systems, such as command and control systems, will also be incorporated into DMT training exercises. Utilities to support the operation of the training exercises (for example, data collection and exercise control functions) will also be incorporated into the exercise configurations.

In HLA terms, each of these components will be incorporated as federates into a DMT federation. Simulators will be adapted to exchange information using the HLA interface specification, as will simulations of the other representations in the simulated training environment. The data available for each federate will be defined in the SOM for that federate, and a FOM will be developed for each exercise, reflecting the specific data exchange requirements for federates needed to meet the specific exercise objectives. Interfaces to live systems, particularly command and control elements, will allow players at their real-world systems to participate in DMT exercises. Finally, depending on the needs of the exercise, data collectors, controllers, or viewers will also be incorporated as participants in the federation. In this way, HLA provides the necessary framework to support the flexible reuse of the key DMT components.

It is unlikely that any application would utilize all the RTI services in the same federation. For DMT applications, there are a subset of HLA features and services which are applicable to their specific needs. Here, we examine two key areas for DMT applications requirements and how HLA can support them.

Simulator and simulation reuse. At the heart of the DMT concept is the general recognition that costly training resources must be viewed as reusable resources that support a variety of training and mission-rehearsal applications. From a technical perspective, this implies that potential DMT resources need to have the capability to join with existing platforms in a common mission-training exercise. To stage such an event, the ability to incorporate the types of training objects needed is required. The special development of all of these training resources—the verification and validation of their representations within DMT—can be prohibitively expensive. Therefore, a principal motivation behind HLA is to make DMT systems more cost-effective by reusing existing resources for virtual training. By making virtual training systems HLA compliant, it is possible to reuse existing platforms for new applications.

This scenario argues for a standard architecture for virtual platform reuse and interoperability. There are also several specific HLA features that particularly address the DMT needs for reuse. Because HLA provides interface standardization with a set of runtime services, it supports incorporation of man-in-the-loop

simulators, pure software representations of systems, and interfaces to live players in a common architecture. All three of these are needed to meet the DMT needs. Also, because HLA does not predefine the data exchanges used by participating federates and federations, it offers the flexibility for users to develop applications tailored to their specific needs. Moreover, because HLA restricts standards to the interfaces, it has little impact on how a federate is constructed, and it leaves open the possibilities for incorporating advances in software representation to meet new and growing DMT needs. Finally, HLA support for distributed applications addresses the geographic dispersion of critical DMT resources. HLA can be implemented in a variety of training environments and lends itself well to the requirements of geographical distribution.

Reconfigurability to meet different and changing user needs. The effectiveness of DMT technologies depends on two factors: ability to access a wide range of potential distributed players, and the ability to flexibly reconfigure DMT resources to meet new and changing needs. HLA ensures these capabilities, as a priority. However, the downside to general purpose capabilities and flexibility is that each application is seen as a new endeavor and the costs of newly designing and developing each application can become prohibitive. So the critical question is how can the DMT community use the general features of HLA to create a flexible, reconfigurable, yet manageable environment for their applications? The answer may be that DMT will be a "persistent federation" that evolves over time to meet changing needs. As a persistent federation, it will employ the FEDEP in an interesting way.

From a process perspective, the FEDEP currently describes a general approach to developing a federation. The FEDEP can be considered in stages across the life cycle of a program. For instance, one could begin the requirements stage of the process and look at the range of DMT requirements for an initial, multiyear phase of the program. These Phase 1 requirements would then be used to define a broad-based federation object model for the set of applications envisioned during this phase. The Phase 1 master FOM could be used to define the superset of data interchange requirements for the full set of federates participating in the federation executions that would support DMT exercises during this period. As Phase 1 progresses, and new requirements emerge, the requirements would be revisited and updated for subsequent phases of the program. In this way, the FEDEP is adapted to support phased requirements with more frequent federation designs and executions within each phase.

From a data interchange specification standpoint, one can envision this master DMT federation object model incorporating the wide range of systems and other training scenario representations needed for a variety of DMT exercises during each phase. This master FOM would be used as a basis for defining the requirements for adaptation of existing simulators, development of new simulators, and for the identification of candidate simulations to be incorporated into DMT federation executions. Defining and promulgating a master DMT FOM will help manage the flexibility HLA offers by imposing data interchange requirements on DMT participants that are tailored to DMT needs. This will allow DMT to recoup some of the perceived out of the box interoperability offered by earlier, less flexible design approaches, while maintaining the capability to customize and extend FOMs as the DMT requirements evolve.

With the availability of a master DMT FOM, other HLA capabilities can further aid DMT in gaining needed flexibility. Experience has shown predefining data exchanges works only to a point. As users employ simulations, new needs and requirements inevitably emerge. The hierarchical structure of HLA FOM allows for FOM extension without requiring change on the part of current federation participants. By the addition of new object subclasses and new object attributes, new federates can send and receive the data they need while the basic set of federates are unaffected by this new information and proceeding without interference.

The master DMT FOM can support federations that use any subset of the information in the master FOM. Federates developed to support the subset appropriate to their capabilities can be readily incorporated into the specific federation and support the federation execution. The use of runtime declaration management services, object management services, and data distribution management services provide the mechanism to limit the amount of data actually exchanged at runtime providing efficiencies in local and destination processing as well as bandwidth. Declaration management provides the means for federates to specify the types of data in the FOM they can send and receive. Object management services support the *source quench* of data available from a federate, but which is not needed by any other member of the federation. Data distribution management services allow for regulating data distributions specified in the FOM by other conditions of relevance to the federation. Together, these services provide the means for a specific DMT federation to efficiently subset a master FOM at runtime.

Conclusion

HLA is a defining architectural standard for DMT systems. However, optimal design of DMT systems should address a variety of factors besides conformance to the standard, for example:

1. What should the DMT system architecture be? How many and what type of processor(s) should be used? In addition to using HLA as a technical blueprint for DMT architectures, the configuration of the training resources should be based on these considerations: What tasks are transferable from the virtual training environment to the real operating environment? How effective are these transfers? How should the training curricula be designed to accommodate virtual training within a broader training program? What do platform operators need to train together in accomplishing team goals? Where and how the DMT resources are located and used? What are the life-cycle costs associated with the DMT resources. Although designing DMT processors through a task analysis and dedication of tasks to separate processors is possible, clear tradeoffs between costs and effectiveness exist. These need to be investigated.

2. Which operating system(s) should be used? Does the operating system offer sufficient real-time control? Is the communication implementation correct (fully match the relevant specification) and efficient? Some operating system implementations of common communication protocols are not particularly efficient, nor do they properly implement the current specifications. Potential choices of systems must be tested across a wide variety of likely operating conditions to make sure these types of issues do not adversely affect a distributed system.

3. Is an RTI with acceptable performance characteristics available on the platform(s) of choice? RTI implementers are free to optimize their code in ways that distinguish that RTI in the marketplace. Users of an RTI must take on the responsibility of characterizing the performance of each perspective RTI implementation to be decided if a given RTI can achieve the required performance levels. DMSO has provided some elementary performance measurement applications, but these should not be viewed as sufficient.

4. How should federations be designed? What are the data transport requirements? Federation designers must carefully consider which transport modes they choose for which data elements in determining both the structure of the federates and their data needs. These decisions must be based on functionality requirements (for example, each copy of the data must be delivered to every subscribing federate); distribution requirements (What is the likely number of federates that will subscribe for this data?); and benchmarking the proposed system components—that is, the proposed RTI implementation in conjunction with proposed platform (hardware and operating system), and in a manner consistent with functionality and distribution requirements.

5. How is the data distribution managed within the RTI? What are the optimization issues? The Data Distribution Management (DDM) services are intended to allow the federation designers to limit how data flows between federates. This is particularly useful when large portions of data sent are irrelevant to some of the federates in the federation. By reducing the arrival of irrelevant data, less processing is required by the federate application, and by the host operating system. RTI implementers are free to use techniques such as IP multicast to assist in implementing DDM. Using IP multicast allows RTI implementers to use available network hardware to reduce delivery of irrelevant data (not subscribed for, or outside the specifications described by the federate via DDM) to federates. For some federations, using DDM to eliminate delivery of irrelevant data to the federate, and even to the host computer, can radically improve system performance. C

References

1. Defense Modeling and Simulation Office. *High Level Architecture Interface Specification 1.3, High Level Architecture Object Model Template 1.3, High Level Architecture Rules 1.3*. Feb. 5, 1998; hla.dmso.mil.
2. Hunt, K., Dahmann, J.S. Sheehan, J., and Lutz, R.P. *Planning for the evolution of automation in the HLA tool suite*. Paper 97S-SIW-067. Simulation Interoperability Workshop. (Orlando, Fla., Mar. 3-7, 1997).
3. IEEE Project Authorization Request (PAR) 1516 Standard for Modeling and Simulation (M&S) High Level Architecture (HLA): Framework and Rules; PAR 1516.1 Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)–Federate Interface Specification; PAR 1516.2 Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Object Model Template (OMT) Specification. Approved by the IEEE New Standards Committee, Dec. 8, 1997.
4. NATO Working Paper AC/323-WP02. *NATO Modeling and Simulation Master Plan*. Feb. 6, 1998.
5. Object Management Group. The Common Object Request Broker *Architecture (CORBA)*. Standards reference, 1997.
6. U. S. Defense Department, Under Secretary of Defense for Acquisition and Technology. USD (A&T) memorandum, Subject: DoD High Level Architecture (HLA) for Simulations. Sept. 10, 1996.

Judith S. Dahmann (jdahmann@dmso.mil) is the chief scientist at the Defense Modeling and Simulation Office, in Alexandria, VA.

James O. Calvin (jcalvin@ll.mit.edu) is an associate group leader of the Distributed Systems Group at Massachusetts Institute of Technology, Lincoln Laboratory, in Lexington, MA.

Richard M. Weatherly (weather@mitre.org) is a chief engineer in the Information Systems and Technology division of The MITRE Corporation, in McLean, VA.