Part 1: Introduction to the High Level Architecture

1.0 What is the High Level Architecture?

The High Level Architecture (HLA) is a standard framework that supports simulations composed of different simulation components. In this module we answer the questions: why do we need the HLA and what are its components?

1.1 Why is the HLA needed?

Many complex simulations involve a combination of simulations of several different types of system with other aspects of the total environment to be simulated. Often simulations of some of these components already exist, having been developed for a different purpose, and they could be used in the new simulation. Unfortunately it is often necessary to make extensive modifications to adapt the component simulation model so that it can be integrated into a new combined simulation. In some cases it may prove easier to implement a completely new simulation of a system component than it is to modify an existing one. In other words traditional simulation models often lack two desirable properties: reusability and interoperability.

Reusability, as the name suggests, means that component simulation models can be reused in different simulation scenarios and applications. Closely related to reusability is the property of interoperability, which means that the reusable component simulations can be combined with other components without the need for re-coding.

Interoperability implies an ability to combine component simulations on distributed computing platforms of different types, often with real-time operation. This approach involves rethinking the ways in which simulation components interact in a traditional single-program, single-computer environment. Rather than a single program executing on a single computer, think of a number of programs executing on distributed computers of different types and interacting with each other through a distributed, real-time operating system.

For example, the development of a new military aircraft and its weapon systems involves a great deal of simulation for different purposes and involving numerous different models. Suppose that a new weapon system is to be installed into an existing aircraft, such as the F-16 fighter. It would be beneficial not to have to develop a completely new simulation from scratch. Ideally one could reuse an existing F-16 simulation with new simulation components representing the new weapon system and simulations of scenarios in which the new system
would be deployed. For all of these components of the complete simulation to function together, possibly distributed over a number of computers of different kinds, they must conform to a standard, which guarantees interoperability. The HLA was developed to respond to the need for a standard of this kind.

1.2 What are the origins of the HLA?

The High Level Architecture (HLA) was developed by the Defense Modeling and Simulation Office (DMSO) of the Department of Defense (DoD) to meet the needs of defense-related projects such as the one outlined above, but it is now increasingly being used in other application areas. The above military scenario can easily be changed to one in which, for example, an existing simulation model of a Boeing 747 is to be used to evaluate a new radar system for terrain monitoring to minimize the danger of crashes in poor visibility in mountainous terrain.

DoD policy is to disseminate information about the HLA as widely as possible, both inside and outside the US, and even to provide free supporting software to help new users to evaluate and use the HLA as easily and inexpensively as possible. Examples of non-military applications that have already used the HLA are traffic simulations and factory production line simulations.

We can consider a complex simulation as a hierarchy of components of increasing levels of aggregation. At the lowest level is the model of a system component. This may be a mathematical model, a discrete-event queuing model, a rule-based model etc. The model is implemented in software to produce a simulation. When this simulation is implemented as part of an HLA-compliant simulation, it is referred to as a federate. HLA simulations are made up of a number of HLA federates and are called federations. There can be multiple instances of a particular type of federate, for example several Boeing 747 simulations or F-16 simulations, in a given federation, and this number can change as the simulation continues. In other words, simulations that use the HLA are modular in nature allowing federates to join and resign from the federation as the simulation executes. Note that federations can include more than simulations. They can also include interfaces to human operators/players, to real hardware and to general software performing functions such as data collection, data analysis, data display etc.
1.3 What does the HLA consist of?

The HLA consists of three components (Figure 1.1):

1. Federation Rules
2. Interface Specification
3. Object Model Template (OMT)

Figure 1.1: Components of the High Level Architecture

Remember: HLA is NOT an implementation, it provides a framework.

1.3.1 THE HLA RULES

At the highest level, the HLA consists of a set of 10 HLA rules which must be obeyed if a federate or federation is to be regarded as HLA compliant. The HLA rules are divided into two groups consisting of 5 rules for HLA federations and 5 rules for HLA federates (Figures 1.2 and 1.3).
Federation Rules

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

Figure 1.2: HLA Rules for Federations

Federate Rules

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

Figure 1.3: HLA Rules for Federates
These rules will be covered in more detail, as we understand more about the use of the HLA. At this stage we will make a quick overview to get an idea of the scope of the rules.

The federation rules establish the ground rules for creating a federation, including documentation requirements (Rule 1), object representation (Rule 2), data interchange (Rule 3), interfacing requirements (Rule 4) and attribute ownership (Rule 5).

The federate rules deal with the individual federates. They cover documentation (Rule 6), control of and transfer of relevant object attributes (Rules 7, 8 and 9), and time-management (Rule 10).

The significance of the rules will become more apparent as we apply them to a specific HLA example.

1.3.2 THE INTERFACE SPECIFICATION

The interface specification defines a standard for a Run-Time Infrastructure (RTI). It is currently proceeding through an IEEE Standard approval process as a standard for distributed simulation. The RTI is software that conforms to the specification but is not itself part of the specification. It provides the software services, which are necessary to support an HLA compliant simulation. Different versions of the RTI are possible. One version is available from DMSO. The current version from DMSO is RTI version 1.3.4 and it can be downloaded free of charge from the DMSO website.

The interface specification identifies how federates will interact with the federation and, ultimately, with one another. An overview of the RTI is given in Figure 1.4 and a list of RTI services appears in Figure 1.5.
Run Time Infrastructure (RTI) Overview

- What is the RTI?
  - Software that provides common services to simulation systems.
  - Implementation of the HLA Interface Specification.
  - An architectural foundation encouraging portability and interoperability.

Figure 1.4: Run-Time Infrastructure (RTI) Overview

RTI Services

- Separate simulation and communication.
- Improves on older standards (e.g., DIS, ALSP).
- Facilitates construction and destruction of federations
- Supports object declaration and management between federates.
- Assists with federation time management.
- Provides efficient communications to logical groups of federates.

Figure 1.5: RTI Services
Interface Specification
Management Areas

- Federation Management
- Declaration Management
- Object Management
- Ownership Management
- Data Distribution Management
- Time Management

Figure 1.6: Interface Specification - Management Areas

Object Model Template

- Object Model Template (OMT)
  - Provides a common framework for HLA object model documentation.
  - Fosters interoperability and reuse of simulations and their components.
- Required Information
  - Object Class Structure Table
  - Object Interaction Table
  - Attribute/Parameter Table
  - FOM/SOM Lexicon
- Optional Information (OMT Extensions)
  - Component Structure Table
  - Associations Table
  - Object Model Metadata

Figure 1.7: Object Model Template
The interface specification is divided into 6 management areas. The areas are listed in Figure 1.6.

They will be explored in more detail as the course proceeds.

1.3.3 THE OBJECT MODEL TEMPLATE

Reusability and interoperability require that all objects and interactions managed by a federate, and visible outside the federate, should be specified in detail and with a common format. The Object Model Template (OMT) provides a standard for documenting HLA Object Model information. An outline of the structure of the OMT appears in Figure 1.7.

The OMT defines the Federation Object Model (FOM), the Simulation (or Federate) Object Model (SOM) and the Management Object Model (MOM). These are summarized in Figure 1.8.

---

**Object Models**

- **Federation Object Model (FOM)**
  - One per federation.
  - Introduces all shared information (e.g., objects, interactions).
  - Contemplates interfederate issues (e.g., data encoding schemes).

- **Simulation Object Model (SOM)**
  - One per federate.
  - Describes salient characteristics of a federate.
  - Presents objects and interactions that can be used externally.
  - Focuses on the federate’s internal operation.

- **Management Object Model (MOM)**
  - Universal definition.
  - Identifies objects and interactions used to manage a federation.

---

**Figure 1.8: Object Model Summary**

The HLA separates data and architecture. It prescribes that OMT objects and interactions defined according to the OMT can be constructed and exchanged with no adjustments to HLA-derived software.
1.4 An example of HLA: HelloWorld

HelloWorld provides a simple example of the use of HLA. It uses federates representing different countries that have an initial population which grows exponentially (i.e. increase in population in each time period is proportional to the current population). Federates publish their current population at each time tick and receive population figures from other federates. Mathematically we would write a simple, first-order differential equation of the form

\[ \frac{dP}{dt} = k \cdot P \] \hspace{1cm} \text{where} \ P=P_0 \ at \ t=0

The equivalent discrete digital representation uses the simple rectangular integration rule to approximate the solution of the differential equation. This gives an equation of the form:

\[ P_{new} = P_{old} + k \cdot P_{old} \cdot \Delta t \]

where
- \( P_{old} \) is the population at time \( t \);
- \( P_{new} \) is the population at time \( t + \Delta t \);
- \( \Delta t \) is the time-step for advancing the simulation;
- \( k \) is the fractional rate of increase of the population.

HelloWorld federates are provided with an initial population and the number of time intervals to simulate. All have the same rate, so the ranking of countries with respect to the size of their populations does not change. Note that when running a federation made up of several federates, the federates are launched one at a time (they are said to join the federation) and if the duration of the simulation is short, the federation will terminate before you have chance to launch further federates. The handout gives detailed instructions for executing the federation.

Assignment

Use the HLA Course Lab Notes to help you execute the basic version of HelloWorld.

Suggested Readings

1. Read High Level Architecture Rules Version 1.3.
