A Distributed Simulation Standard: High Level Architecture (HLA) (Part 8)

Dr. Çağatay ÜNDEĞER

Öğretim Görevlisi
Bilkent Üniversitesi Bilgisayar Mühendisliği Bölümü

e-mail : cagatay@undeger.com
cagatay@cs.bilkent.edu.tr

A Distributed Simulation Standard: High Level Architecture (HLA) (Outline)

• Introduction to Distributed Simulations
• Introduction to HLA
• HLA Interface Specification:
  – Federation Management
  – Declaration Management
  – Object Management
• Conclusion
Distributed Simulation

- An integrated simulation that is **partitioned into a number of smaller simulations** over different computational units (e.g. processors, computers).
- Provides **higher scalability** and multi user interaction.
- A system, whose performance improves proportional to the computational capacity added, is said to be a **scalable system**.
High Level Architecture (HLA)

- An IEEE (1516) Distributed Modeling & Simulation Standard
- Developed By DMSO
  - Current technology was not providing tools necessary to achieve DoD M&S Master Plan.
  - A Standard was Needed For the Interoperability of Developed Simulations.

What is HLA?

- A service that provides:
  - Major functional elements,
  - Interfaces,
  - Design rules,
  - A common framework for communication of simulations.
The Structure of HLA

Federation

Simulations (Federates)

Usually a Centralized Approach
All Simulations Communicate via RTI

RUN TIME INFRASTRUCTURE (RTI)

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

Defining the HLA

- The HLA is composed of three elements:
  - HLA Rules:
    - A set of rules which must be followed to achieve proper interaction of simulations in a federation
  - Interface Specification:
    - Definition of the interface functions between the RTI and the simulations subject to the HLA
  - Object Model Template (OMT):
    - The prescribed common method for recording the information contained in the required HLA Object Model for each federation and simulation
HLA Object Models

FOM: Federation object model
SOM: Simulation object model

FOM of Federation
e.g. A Flight simulation

SOM of Federate
e.g. A F14 Simulator
SOM of Federate
e.g. Weapon Trajectory Manager

RTI

OMT, documentation of the system, is saved into a data storage

HLA Rules

• Federation Rules:
  – Federations shall have an HLA Federation Object Model (FOM), documented in accordance with the HLA Object Model Template.
  – In a federation, all representation of objects in the FOM shall be in the federates, not in the runtime infrastructure (RTI).
  – . . . . .

• Federate Rules:
  – Federates shall have an HLA Simulation Object Model (SOM), documented in accordance with the HLA Object Model Template.
  – Federates shall be able to update and/or reflect any attributes of objects in their SOM and send and/or receive SOM object interactions externally, as specified in their SOM.
  – . . . . .
HLA Interface Specification

<table>
<thead>
<tr>
<th>Category</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Federation Management</td>
<td>Create and delete federation executions</td>
</tr>
<tr>
<td></td>
<td>Join and resign federation executions</td>
</tr>
<tr>
<td></td>
<td>Control checkpoint, pause, resume, restart</td>
</tr>
<tr>
<td>Declaration Management</td>
<td>Establish intent to publish and subscribe to object attributes and interactions</td>
</tr>
<tr>
<td>Object Management</td>
<td>Create and delete object instances</td>
</tr>
<tr>
<td></td>
<td>Control attribute and interaction publication</td>
</tr>
<tr>
<td></td>
<td>Create and delete object reflections</td>
</tr>
<tr>
<td>Ownership Management</td>
<td>Transfer ownership of object attributes</td>
</tr>
<tr>
<td>Time Management</td>
<td>Coordinate the advance of logical time and its relationship to real time</td>
</tr>
<tr>
<td>Data Distribution Management</td>
<td>Supports efficient routing of data</td>
</tr>
</tbody>
</table>

Federation Management

- Federation management refers to the creation, dynamic control, modification, and deletion of a federation execution.

Basic steps of federation execution
Overview of Federate To RTI Relationship

Federation Management

Initialization/Finalization of a Federate

- CreateFederationExecution
- JoinFederationExecution
- ...
- Normal federate execution....
- ...
- ResignFederationExecution
- DestroyFederationExecution

Initialization of a federate

Finalization of a federate
Declaration Management

- Federates use Declaration Management services to declare their intention;
  - To generate information, and
  - To receive information.
  - A joined federate shall invoke appropriate Declaration Management services

  - Before it may register object instances, update instance attribute values, and send interactions.
Declaration Management

• PublishObjectClass
• UnpublishObjectClass

Declarations which classes it is planning to use in order to create object instances of these classes.

• SubscribeObjectClassAttributes
• UnsubscribeObjectClass

Declarations which classes (and their attributes) it is planning to use in order to receive information of object instances of these classes.

Object Management

• Deals with;
  – The registration, modification, and deletion of object instances, and
  – The sending and receipt of interactions.
• Object instance discovery is the prime concept in this service group.
Object Management

• When a joined federate either;
  – Uses the *Register Object Instance* service to register an object instance or
  – Receives an invocation of the *Discover Object Instance* to discover an object instance,
• That object instance becomes known to the joined federate.

---

Object Management

**Publishing Services**
- RegisterObjectInstance
- UpdateAttributeValues
- DeleteObjectInstance

**Subscribing Services**
- DiscoverObjectInstance
- ReflectAttributeValues
- RemoveObjectInstance

**RUN TIME INFRASTRUCTURE (RTI)**
Support Services

• GetObjectClassHandle
  • All the classes defined in the FOM has a unique name and handle.
  • This function returns the handle given the name of the class.
    e.g. "Aircraft.bomber.F15"

• GetAttributeHandle
  • All the attributes of classes defined in the FOM has a unique name and handle.
  • This function returns the handle given the name of the attribute.
    e.g. "Aircraft.bomber.F15.x"

Class Hierarchy and Publish/Subscribe Mechanism

• Classes are defined as a hierarchy such as:
  • Aircraft [ Attributes: Type,X,Y,Z,H,P,R,V ]
    • Bomber [ Attributes: NumOfFullBombs ]
      – F15 [ Attributes: WingsOn ]
    • Fighter [ Attributes: NumOfFullRightPylons, ... ]
      – F14 [ Attributes: WingsOn, GearsOn ]
      – F16 [ Attributes: FiringGun ]
  
• Federates can publish and/or subscribe to one or more arbitrary classes and attributes on the hierarchy.
• RTI should filter the data and send to the federates only what they are interested in.
Example: Discovering Instances

Federation Execution Data File (FED File)

(FED
(Federation FederationName)
(FEDversion v1.3)
(spaces )
(objects
(class ObjectRoot
  (attribute privilegeToDelete reliable timestamp)
  (class RTIprivate)
(class Aircraft
  (attribute X best_effort receive)
  (attribute Y best_effort receive)
  (attribute Z best_effort receive)
(class Fighter
  (attribute NumOfFullRightPylons best_effort receive)
  (attribute NumOfFullLeftPylons best_effort receive)
  (class F16
    (attribute FiringGun best_effort receive)
  )
)
...
)

FED file defines:

classes,
attributes of classes,
interactions,
parameters of interactions,
routing spaces,
dimensions of spaces,
...
of a FOM
Conclusion

• COTS RTI implementations have a Centralized Architecture.
• If RTI fails,
  – Whole simulation will fail.
• If a federate fails,
  – The simulation continues unless a synchonized timing approach is used.