The High Level Architecture

Introduction

Richard M. Fujimoto
Professor

Computational Science and Engineering Division
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0765, USA

http://www.cc.gatech.edu/~fujimoto/

Outline

- High Level Architecture (HLA): Background
- Rules
- Interface Specification
  - Overview
  - Class Based Subscription
  - Attribute updates

HLA: Motivation

Department of Defense plagued by "stovepipe simulations": individual simulations designed and tailored for a specific application

- Not easily adapted for other uses, resulting in limited software reuse, much duplication of effort
- Cannot easily exploit capabilities developed in other DoD modeling and simulation programs

Goal of the High Level Architecture: define a common simulation infrastructure to support interoperability and reuse of defense simulations

- Analytic simulations (e.g., wargames)
- Training (platform-level, command-level)
- Test and Evaluation

Distributed Simulation in the DoD

- SIMNET (SIMulator NETworking) (1983-89)
  - DARPA and U.S. Army project
  - networked interactive combat simulators
  - tens to a few hundreds of simulators
- DIS (Distributed Interactive Simulation) (1990-96)
  - rapid expansion based on SIMNET success
  - tens of thousands of simulated entities
  - IEEE standard
- Aggregate Level Simulation Protocol (ALSP) (late 1980's and 1990's)
  - application of the networked simulations concept to wargaming models

HLA Development Process

- 10/93-1/95:three architecture proposals developed in industry
- 3/95: DMSO forms the Architecture Management Group (AMG)
- 3/95-8/96: development of baseline architecture
  – AMG forms technical working groups (IFSpec, time management, data distribution management)
  – Run-Time Infrastructure (RTI) prototypes
  – prototype federations: platform level training, command level training, engineering test and evaluation, analytic analysis
- 8/96-9/96: adoption of the baseline architecture
  – approval by AMG, Executive Council for Modeling and Simulation (EXCIMS), U.S. Under Secretary of Defense (Acquisition and Technology)
  – 10 September, 1996: Baseline HLA approved as the standard technical architecture for all U.S. DoD simulations
- 9/96-present: continued development and standardization
  – Varying levels of adoption
  – Commercialization of RTI software
  – Standardization (IEEE 1516)

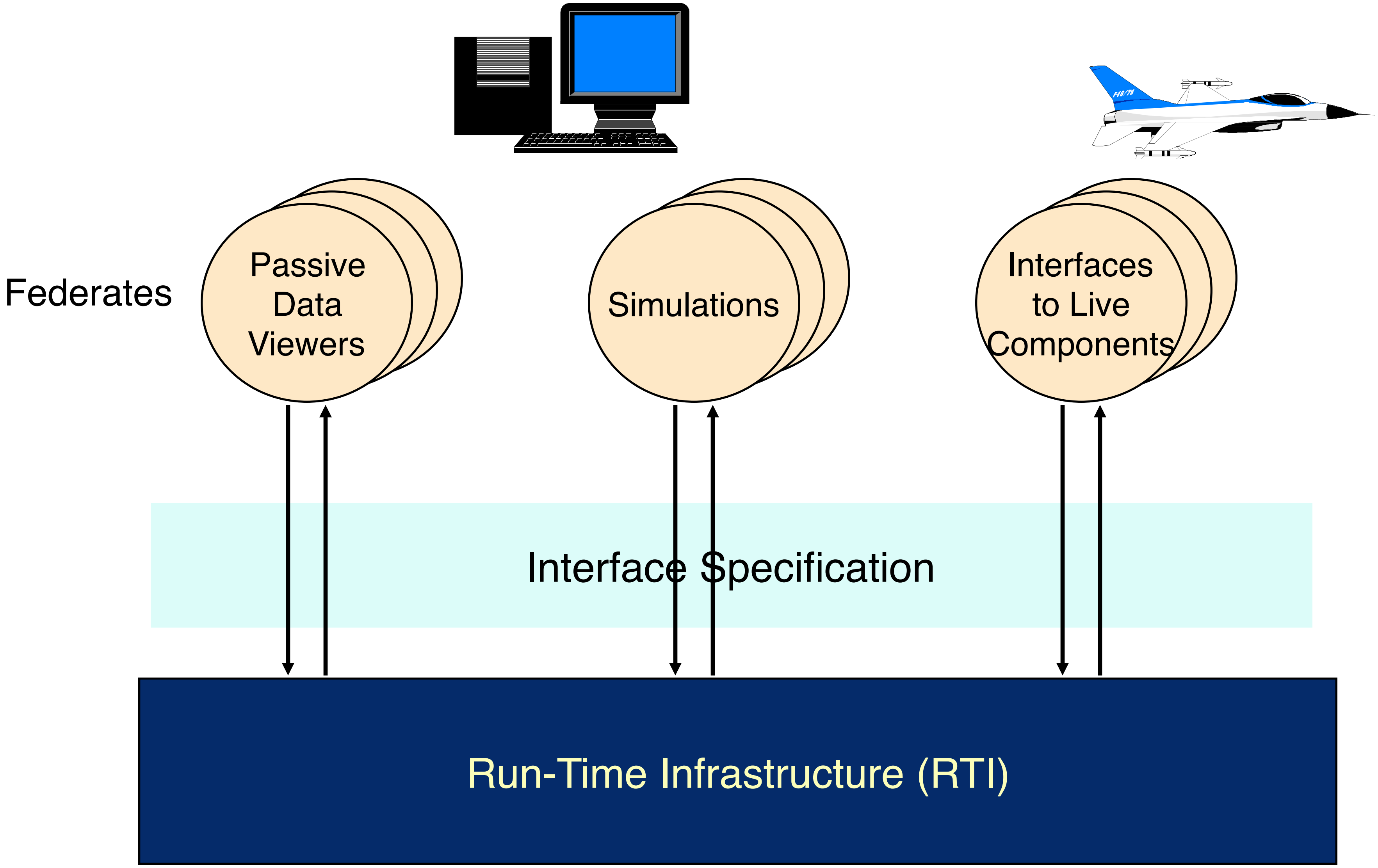
High Level Architecture (HLA)

- based on a composable “system of systems” approach
  - no single simulation can satisfy all user needs
  - support interoperability and reuse among DoD simulations
- *federations* of simulations (*federates*)
  - pure software simulations
  - human-in-the-loop simulations (virtual simulators)
  - live components (e.g., instrumented weapon systems)

The HLA consists of

- **Rules** that simulations (federates) must follow to achieve proper interaction during a federation execution
- **Object Model Template (OMT)** defines the format for specifying the set of common objects used by a federation (federation object model), their attributes, and relationships among them
- **Interface Specification (IFSpec)** provides interface to the *Run-Time Infrastructure* (*RTI*), that ties together federates during model execution

An HLA Federation



Federation Rules

1 Federations shall have an HLA Federation Object Model (FOM), documented in accordance with the HLA Object Model Template (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 joined federates shall occur via the RTI.

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

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

Federate Rules

6 Federates shall have an HLA Simulation Object Model (SOM), documented in accordance with the HLA Object Model Template (OMT).

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

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 SOM.

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.

Interface Specification

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

Message Passing Alternatives

- Traditional message passing mechanisms: Sender explicitly identifies receivers
  - Destination process, port, etc.
  - Poorly suited for federated simulations
- Broadcast
  - Receiver discards messages not relevant to it
  - Used in SIMNET, DIS (initially)
  - Doesn't scale well to large federations
- Publication / Subscription mechanisms
  - Analogous to newsgroups
  - Producer of information has a means of describing data it is producing
  - Receiver has a means of describing the data it is interested in receiving
  - Used in High Level Architecture (HLA)

A Typical Federation Execution

initialize federation

- Create Federation Execution (Federation Mgt)
- Join Federation Execution (Federation Mgt)

declare objects of common interest among federates

- Publish Object Class Attributes (Declaration Mgt)
- Subscribe Object Class Attributes (Declaration Mgt)

exchange information

- Update/Reflect Attribute Values (Object Mgt)
- Send/Receive Interaction (Object Mgt)
- Time Advance Request, Time Advance Grant (Time Mgt)
- Request Attribute Ownership Assumption (Ownership Mgt)
- Send Interaction with Regions (Data Distribution Mgt)

terminate execution

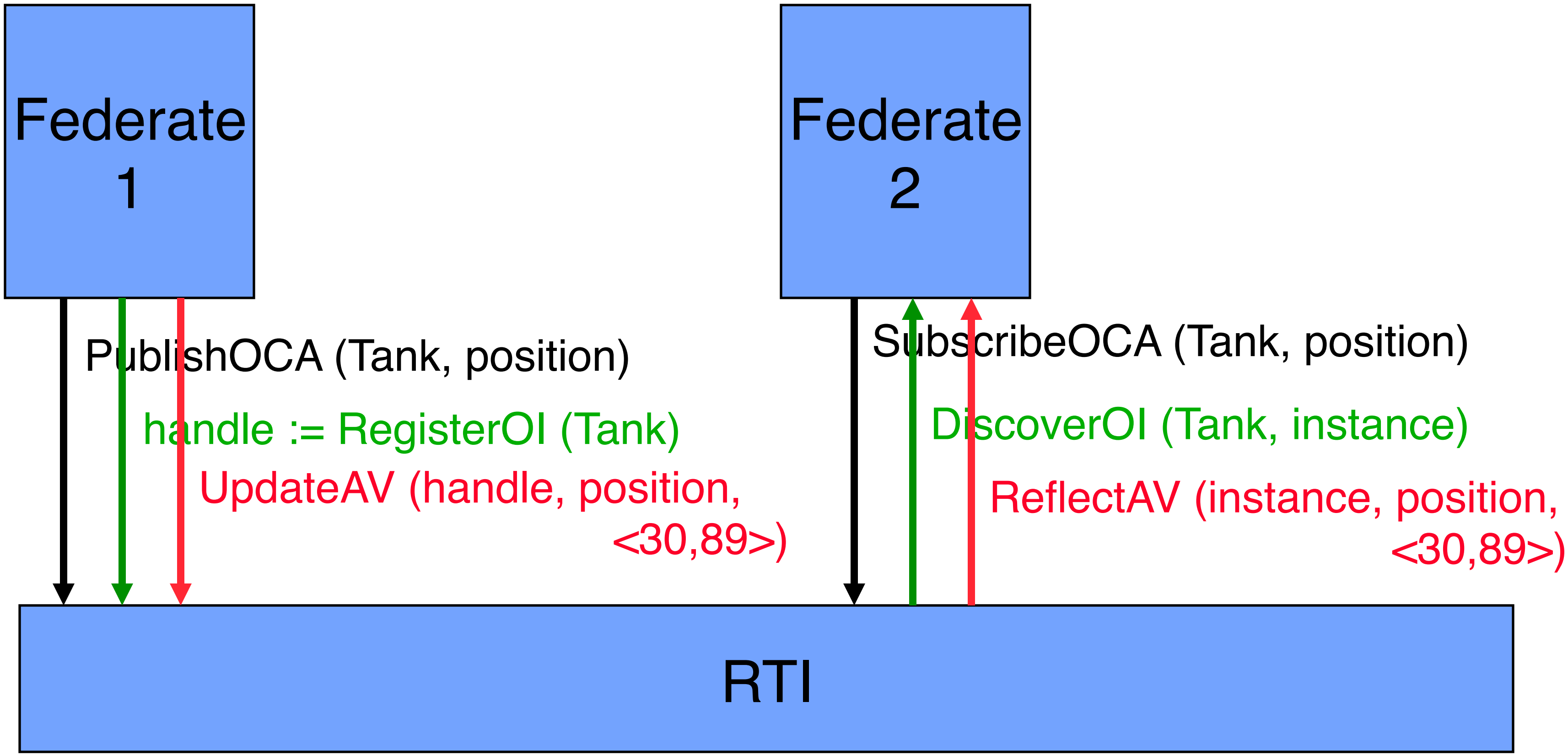
- Resign Federation Execution (Federation Mgt)
- Destroy Federation Execution (Federation Mgt)

Class-Based Data Distribution

- Federation Object Model (FOM) defines type of information transmitted among federates
  - Object classes (e.g., tank)
  - Attributes (e.g., position, orientation of turret)
- A few key primitives (Federate/RTI interface)
  - Publish Object Class Attributes: Called by a federate to declare the object classes and attributes it is able to update
  - Subscribe Object Class Attributes: Declare the object classes and attributes that the federate is interested in receiving
  - Register Object Instance: Notify RTI an *instance* of an object has been created within the federate
  - Discover Object Instance*: Notify federate an instance of an object of a subscribed class has been registered
  - Update Attribute Values: notify RTI one or more attributes of an object has been modified
  - Reflect Attribute Values*: notify federate attributes to which it has subscribed have been modified

* Denotes callback from RTI to federate

Example



OCA = Object Class Attributes

OI = Object Instance

AV = Attribute Values

Object Model Template

- Data meta-model that describes the information passed among federates
- Tabular representation of objects, attributes, and other information
- Object models describe:
  - The set of shared objects chosen to represent the real world for a planned simulation or a 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
  - The key models and algorithms used in representing the objects

HLA Object Models

- Simulation Object Model (SOM)
  - One defined per simulator (*federate*)
  - Describes objects, attributes and interactions in a particular simulation which *can* be used externally in a federation
- Federation Object Model (FOM)
  - One defined per *federation*
  - A description of all shared information (objects, attributes, associations, and interactions) essential to a particular federation
- Object Model Template (OMT)
  - Provides a common framework for HLA object model documentation
  - Fosters interoperability and reuse of simulations and simulation components via the specification of a common representational framework
- Not an object-oriented programming language

Object Model Template

- Object Class Structure Table
  - specifies the object class hierarchy
- Attribute Table
  - describes object attributes
  - name, type, units, resolution, accuracy, etc.
- Interaction Class Structure Table
  - class hierarchy for interactions
- Parameter Table
  - specifies parameters for interactions
  - name, type, units, resolution, accuracy, etc.
- FOM/SOM Lexicon
  - defines terms used in the other tables

Sample Class Structure Table

Table 5—Object class structure table example

HLA object Root (N)	Customer (PS)				
	Bill (PS)				
	Order (PS)				
	Employee (N)	Greeter (PS)			
		Waiter (PS)			
		Cashier (PS)			
		Dishwasher (PS)			
		Cook (PS)			
	Food (S)	MainCourse (PS)			
		Drink (S)	Water (PS)		
			Coffee (PS)		
			Soda (PS)		
		Appetizers (S)	Soup (S)	ClamChowder (PS)	Manhattan (P)
					NewEngland (P)
				BeefBarley (PS)	
			Nachos (PS)		
		Entree (S)	Beef (PS)		
			Chicken (PS)		
			Seafood (S)	Fish (PS)	
				Shrimp (PS)	
				Lobster *[1] (PS) *[2]	
			Pasta (PS)		

(S) = Subscribe (PS) = Publish and Subscribe

Source: IEEE Std P1516.2/D4, HLA Standard, January 2008

Sample Attribute Table

Table 9—Attribute table example

Object	Attribute	Datatype	Update type	Update condition	D/A	P/S	Available dimensions	Transport-ation	Order
HLAobject Root	HLA privilege ToDelete Object	HLAtoken	NA	NA	N	N	NA	HLAreliable	Time Stamp
Employee	PayRate	DollarRate	Conditional	Merit increase *[3,4]	DA	PS	NA	HLAreliable	Time Stamp
	YearsOf Service	Years	Periodic	1/year *[3]	DA	PS	NA	HLAreliable	Time Stamp
	Home Number	HLAASCII string	Conditional	Employee request	DA	PS	NA	HLAreliable	Time Stamp
	Home Address	Address Type	Conditional	Employee request	DA	PS	NA	HLAreliable	Time Stamp
Employee. Waiter	Efficiency	Waiter Value	Conditional	Performance review	DA	PS	NA	HLAreliable	Time Stamp
	Cheerful-ness	Waiter Value	Conditional	Performance review	DA	PS	NA	HLAreliable	Time Stamp
	State	Waiter Tasks	Conditional	Work flow	DA	PS	NA	HLAreliable	Time Stamp
Food.Drink	Number Cups	DrinkCount	Conditional	Customer request	N	PS	BarQuantity	HLAreliable	Time Stamp
Food.Drink. Soda	Flavor	FlavorType	Conditional	Customer request	N	PS	SodaFlavor, BarQuantity	HLAreliable	Time Stamp
Note	NA								

Source: IEEE Std P1516.2/D4, HLA Standard, January 2008

Summary

- The High Level Architecture is an example of an approach for realizing distributed simulations
- HLA Rules define general principles that pervade the entire architecture
- HLA Interface Specification defines a set of run-time services to support distributed simulations
- Data distribution is based on a publication / subscription mechanism