SEN9110 Simulation Masterclass Lecture 8: High Level Architecture

Prof.dr.ir Alexander Verbraeck
a.verbraeck@tudelft.nl

Brightspace: SEN9110

The High Level Architecture (IEEE 1516) standard

The Society for Computer Simulation

McLeod Institute of Simulation Sciences
California State University, Chico

Slides partly based on Sep '99 presentation from Dr. Judith S. Dahmann, (Chief Scientist DMSO) High Level Architecture for Simulation
and from the HLA course by teh McLeod Institute of Simulation Sciences
California State University, Chico. Prof. Roy Crosbie

What is HLA?

- HLA is a flexible, reusable software architecture for **creating component-based distributed simulations**
- Developed to allow **reuse** of Defense simulations in different applications
- Conceived as **general purpose** architecture applicable beyond defense
- Became an **industry** standard via IEEE standard 1516

Prototyping During HLA Baseline Development

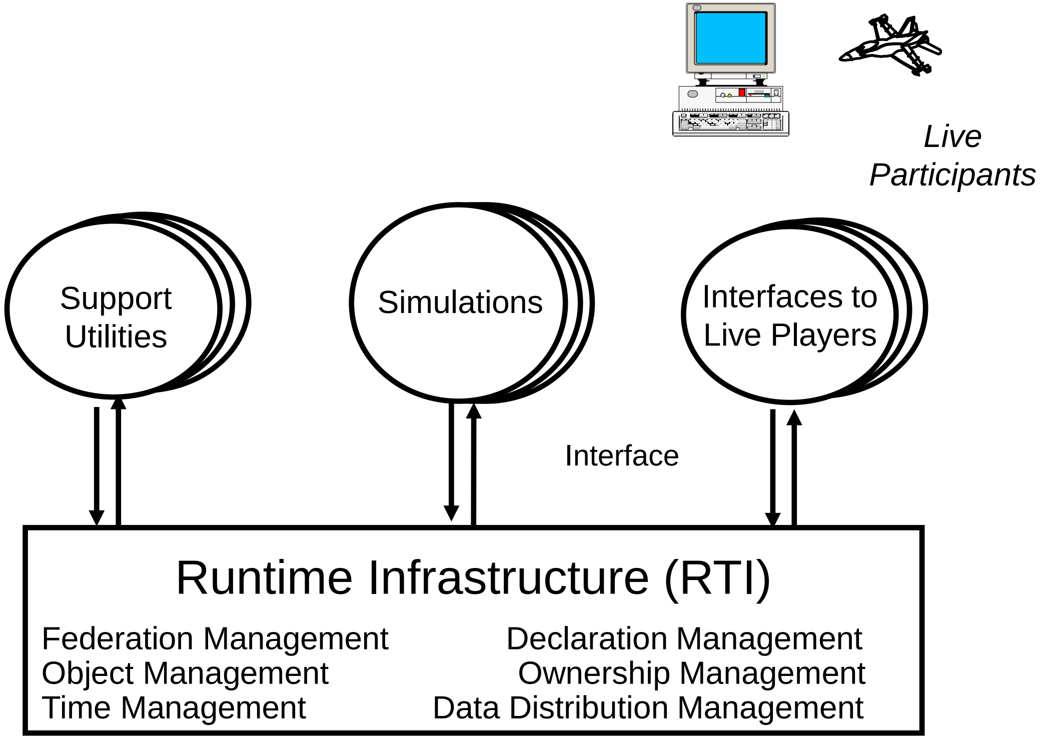
- Over 25 different simulations
- One Runtime Infrastructure (RTI) implementation
- Training, analysis, and acquisition applications
- Unit, platform, and weapon system component level granularity
- Hardware-in-the-loop, human-in-the-loop, and closed-form simulations (live, virtual, and constructive)
- Both real-time and fast-as-possible discrete event simulations
- Both classified and unclassified federations
- Local and wide area networks Run on Sun, Silicon Graphics, HP, and IBM workstations

HLA Standardization Beyond DoD

- Object Management Group
  - Adopted HLA Runtime Services as Distributed Simulation Facility in November 1998
- NATO
  - Adopted HLA in NATO M&S Master Plan in December 1998
- IEEE
  - Standardization process began in May 1998
  - HLA IEEE 1516 is an industry standard

The High Level Architecture (HLA)

- Architecture calls for a federation of simulations
- Architecture specifies
  - Ten Rules which define relationships among federation components
  - An Object Model Template which specifies the form in which simulation elements are described
  - An Interface Specification which describes the way simulations interact during operation



The HLA is not the RTI; the HLA says there will be an RTI that meets HLA requirements but it doesn't specify a particular software implementation

Some Terminology

- **Federation**: a set of simulations, a common federation object model, and supporting RTI, that are used together to form a larger model or simulation

- **Federate**: a member of a federation; one simulation
  - Could represent one platform, like a cockpit simulator
  - Could represent an aggregate, like an entire national simulation of air traffic flow

- **Federation Execution**: a session of a federation executing together

Some More Terminology

- **Object**: An entity in the domain being simulated by a federation that
  - Is of interest to more than one federate
  - Is handled by the Runtime Infrastructure
- **Interaction**: a non-persistent, time-tagged event generated by one federate and received by others (through RTI)
- **Attribute**: A named datum (defined in Federation Object Model) associated with each instance of a class of objects
- **Parameter**: A named datum (defined in Federation Object Model) associated with each instance of a class of interactions

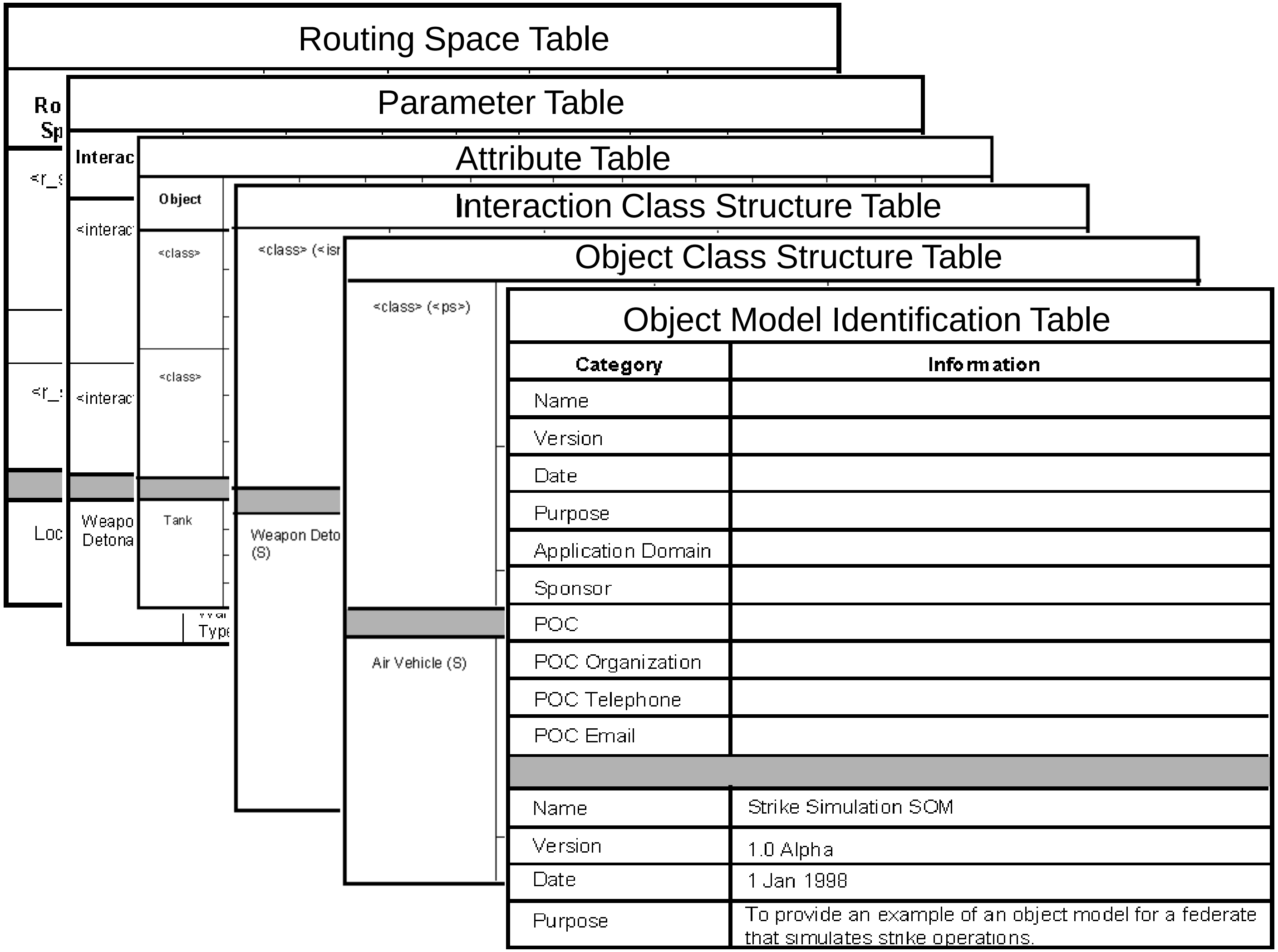
HLA Object Models and OMT

- **Object Model Template** (OMT)
  - Provides a common framework for HLA object model documentation
  - Fosters interoperability and reuse of simulations via the specification of a common representational framework

- **Federation Object Model** (FOM)
  - A description of all shared information (objects, attributes, and interactions) essential to a particular federation

- **Simulation Object Model** (SOM)
  - Describes objects, attributes and interactions in a particular simulation which can be used externally in a federation

Object Model Template



Routing Space Table

Parameter Table

Attribute Table

Interaction Class Structure Table

Object Class Structure Table

Object Model Identification Table

Category	Information
Name	
Version	
Date	
Purpose	
Application Domain	
Sponsor	
POC	
POC Organization	
POC Telephone	
POC Email	
Name	Strike Simulation SOM
Version	1.0 Alpha
Date	1 Jan 1998
Purpose	To provide an example of an object model for a federate that simulates strike operations.

Sample Object Model Identification Table

Object Model Identification Table

Category	Information
Name	Restaurant SOM
Version	1.0 Alpha
Date	1 Jan 1998
Purpose	Provide Object Model for Restaurant Federate
Application Domain	Restaurant Operations
Sponsor	Federated Foods
POC	Mr. Joseph Smith
POC Organization	Joe’s Place
POC Telephone	(977) 555 1234
POC Email	Smithj@fedfoods.com

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

Sample Class Structure Table

Object Class Structure Table				
Customer (PS)				
Bill (PS)				
Order (PS)				
Employee (S)	Greeter (PS)			
	Waiter (PS)			
	Cashier (PS)			
Food (S)	Main_Course (PS)			
	Appetizer (S)	Soup (S)	Clam_Chowder (S)	Manhattan (PS)
				New_England (PS)
			Beef_Barley (PS)	
		Salad (PS)		
	Entrée (S)	Seafood (S)	Fish (PS)	
			Shrimp (PS)	
		Pasta (PS)		

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

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

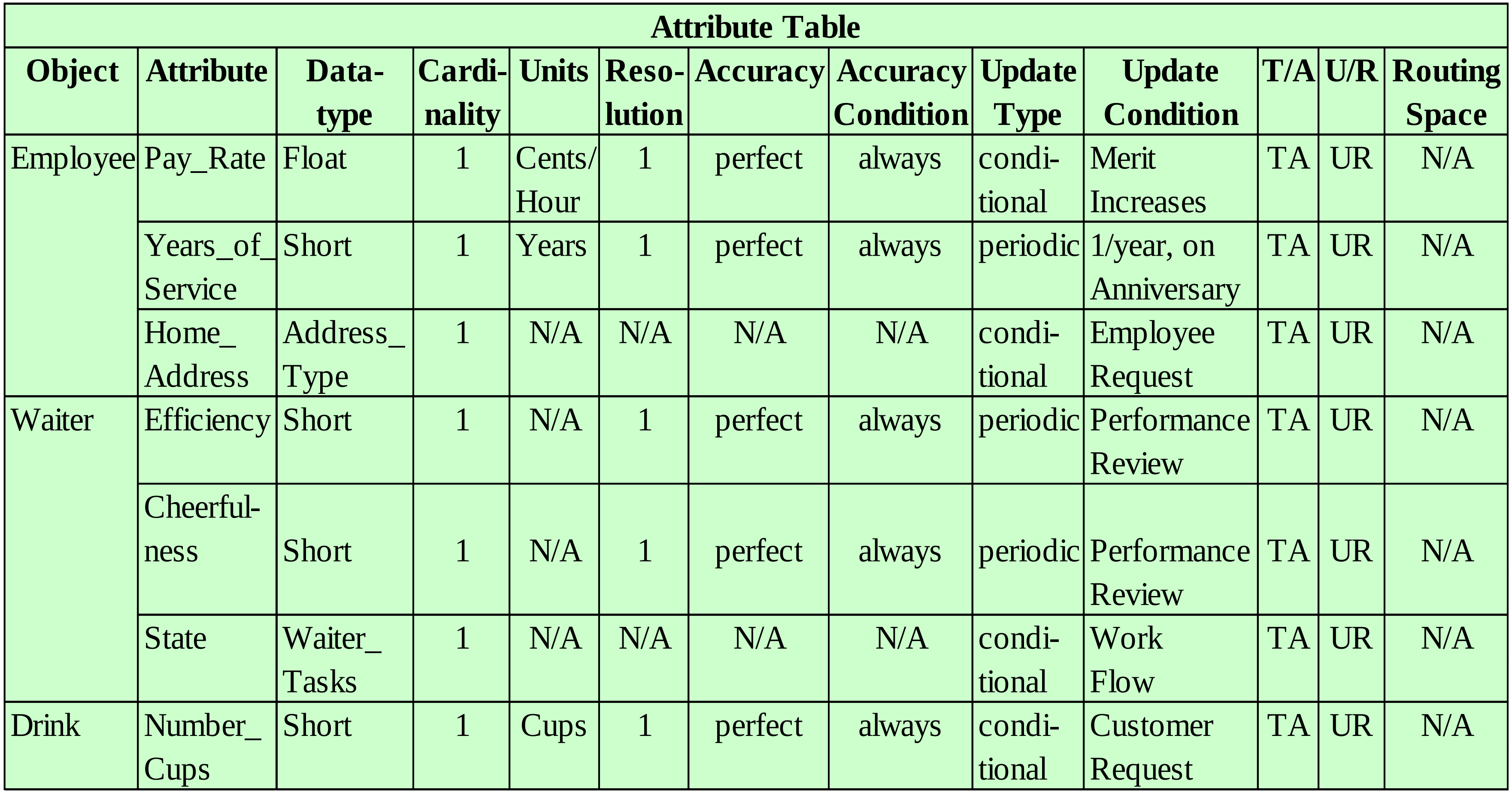
Sample Interaction Structure Table

Interaction Class Structure Table		
Customer_ Employee_ Transactions (I)	Customer_Seated (IS)	
	Order_Taken (I)	Order_Taken_ From_Kids_Menu (I)
		Order_Taken_ From_Adult_Menu (I)
	Food_Served (I)	Drink_Served (I)
		Appetizer_Served (I)
		Main_Course_Served (I)
		Dessert_Served (I)
	Customer_Pays (I)	Pay_Bill_by_
		Credit_Card (I)

(I) = Initiates (S) = Senses (R) = Reacts (N) = Neither initiates, senses, or reacts

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

Sample Attribute Table



Attribute Table

Object	Attribute	Data-type	Cardinality	Units	Resolution	Accuracy	Accuracy Condition	Update Type	Update Condition	T/A	U/R	Routing Space
Employee	Pay_Rate	Float	1	Cents/Hour	1	perfect	always	conditional	Merit Increases	TA	UR	N/A
	Years_of_Service	Short	1	Years	1	perfect	always	periodic	1/year, on Anniversary	TA	UR	N/A
	Home_Address	Address_Type	1	N/A	N/A	N/A	N/A	conditional	Employee Request	TA	UR	N/A
Waiter	Efficiency	Short	1	N/A	1	perfect	always	periodic	Performance Review	TA	UR	N/A
	Cheerfulness	Short	1	N/A	1	perfect	always	periodic	Performance Review	TA	UR	N/A
	State	Waiter_Tasks	1	N/A	N/A	N/A	N/A	conditional	Work Flow	TA	UR	N/A
Drink	Number_Cups	Short	1	Cups	1	perfect	always	conditional	Customer Request	TA	UR	N/A

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

Sample Parameter Table

Parameter Table								
Interaction	Parameter	Data-type	Cardinality	Units	Resolution	Accuracy	Accuracy Condition	Routing Space
Main_Course_Served	Temperature_OK	Temp_Type	1	N/A	N/A	N/A	N/A	N/A
	Accuracy_OK	Accur_Type	1	N/A	N/A	N/A	N/A	
	Timeliness_OK	Boolean	1	N/A	1	perfect	always	

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

Sample Routing Space Table

Routing Space Table					
Routing Space	**Dimension**	**Dimension Type**	**Dimension Range/Set**	**Range/Set Units**	**Normalization Function**
Bar_Order	Soda_flavor	Flavor_Type	Cola, Orange, Root Beer	N/A	linear_enumerated (Flavor)
	Bar_Quantity	short	[1-25]	N/A	linear (Number_Cups)
Server_Order	Waiter_ID	short	[1-20]	N/A	linear (Waiter_ID)

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

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.

- 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.

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

Interface Specification

- Provides a specification of the functional interfaces between federates and the RTI
  - Interfaces are divided into six service groups
- Each service specification includes:
  - Name and Descriptive Text
  - Supplied Arguments
  - Returned Arguments
  - Pre-conditions
  - Post-conditions
  - Exceptions
  - Related Services
- Application Programmer Interfaces (APIs)
  - CORBA IDL, C++, Ada, Java, many others

What does the Interface Specification include?

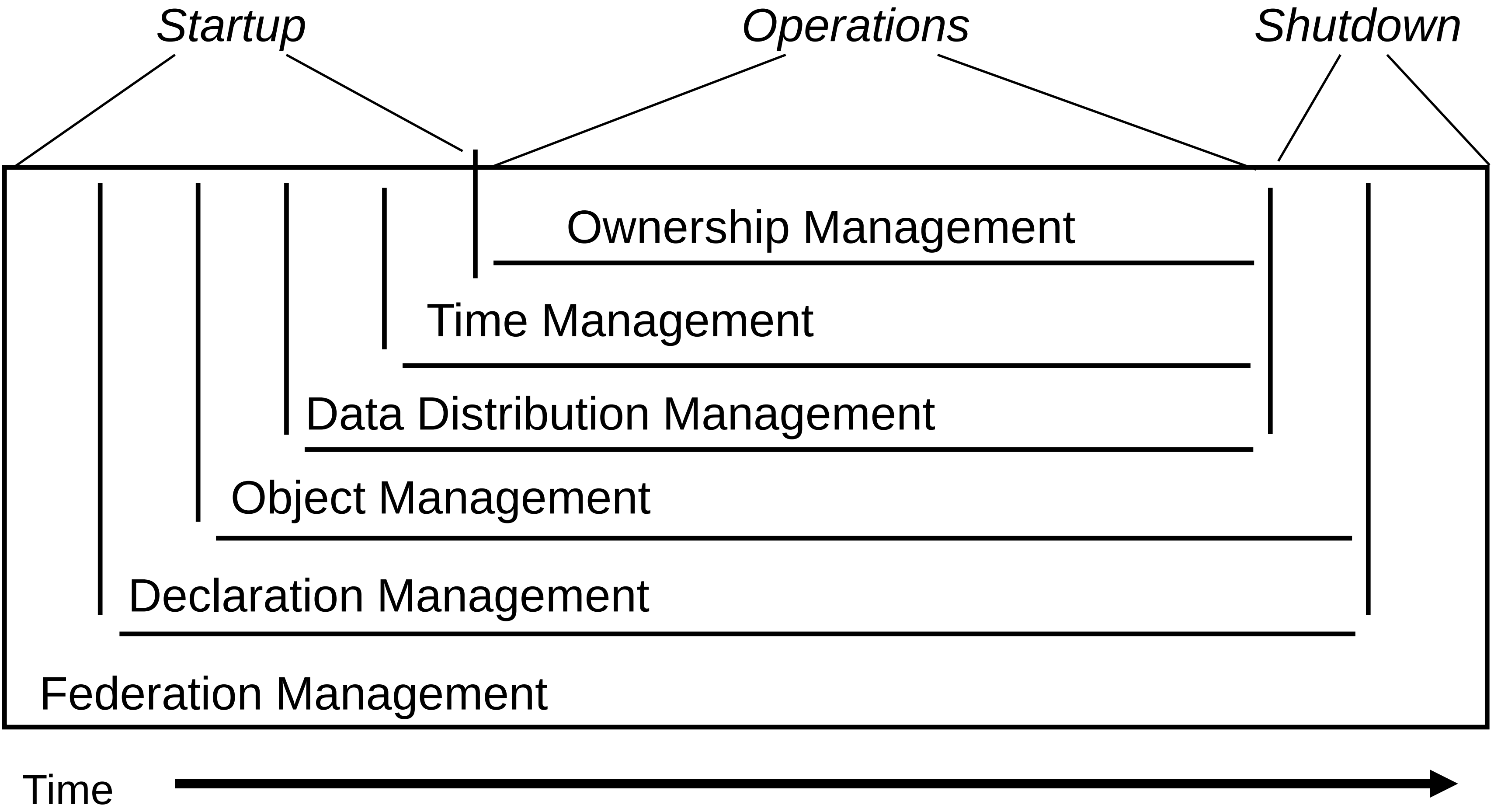
- Six HLA RTI Service Groups
  - Federation Management (20 services)
  - Declaration Management (12 services)
  - Object Management (17 services)
  - Ownership Management (16 services)
  - Time Management (23 services)
  - Data Distribution Management (13 services)

- The Interface Specification also includes:
  - Support Services (29 services)
  - Management Object Model
  - Federation Execution Data (FED)
  - Application Programmers Interfaces (APIs)
  - Harel state charts

HLA RTI Services Categories

Category	Functionality
Federation Management	Create and delete federation executions Join and resign federation executions Control checkpoint, synchronization
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 Mgmt	Supports efficient routing of data

HLA RTI Services over the Life of a Federation



Startup

Operations

Shutdown

Ownership Management

Time Management

Data Distribution Management

Object Management

Declaration Management

Federation Management

Time

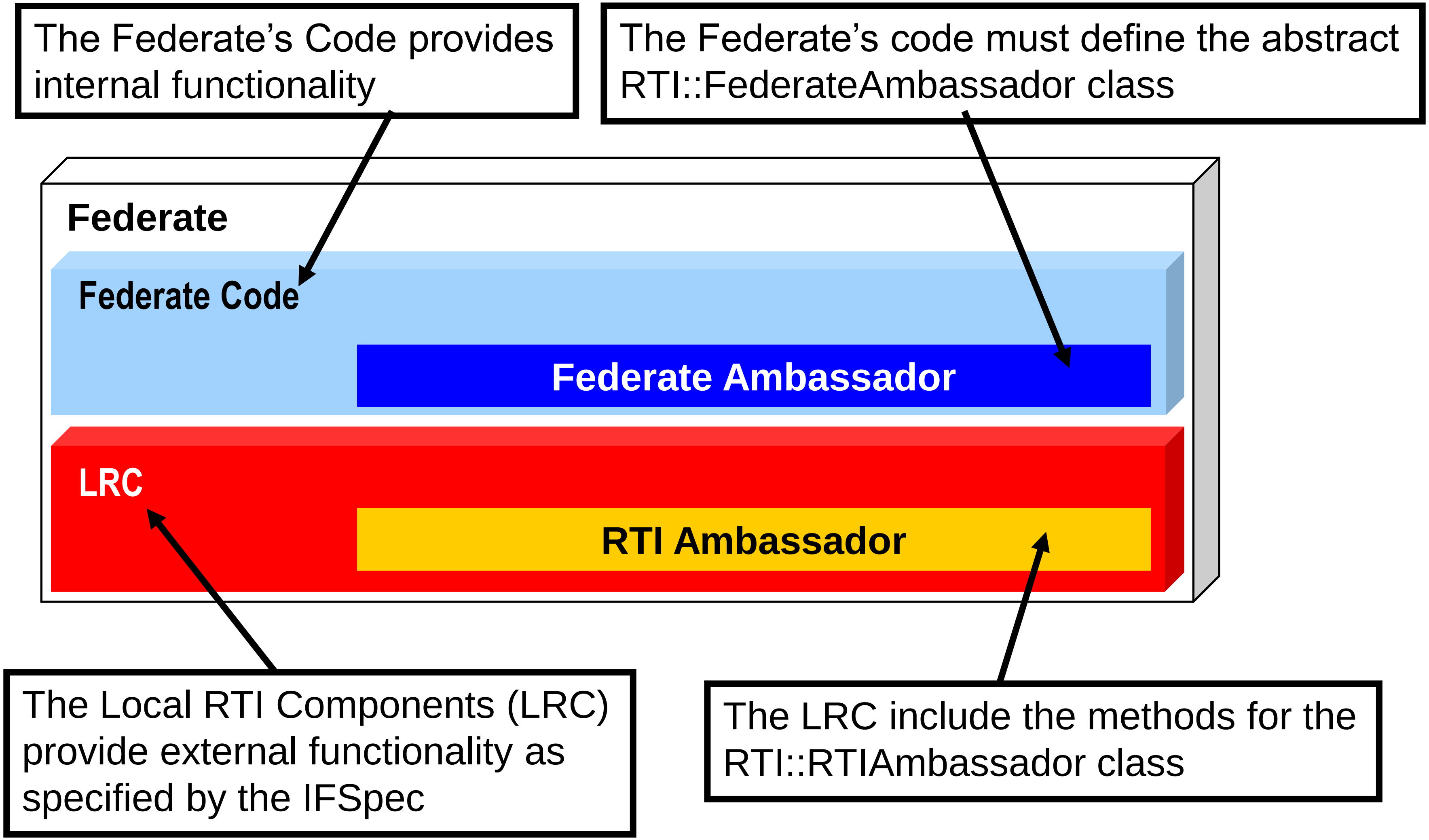
Federation Rules

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 representation of objects in the FOM 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, federates shall interact with the runtime infrastructure (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.

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 attributes of objects in their SOM and send and/or receive SOM object 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 (e.g., thresholds) 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.

What is in a Federate?



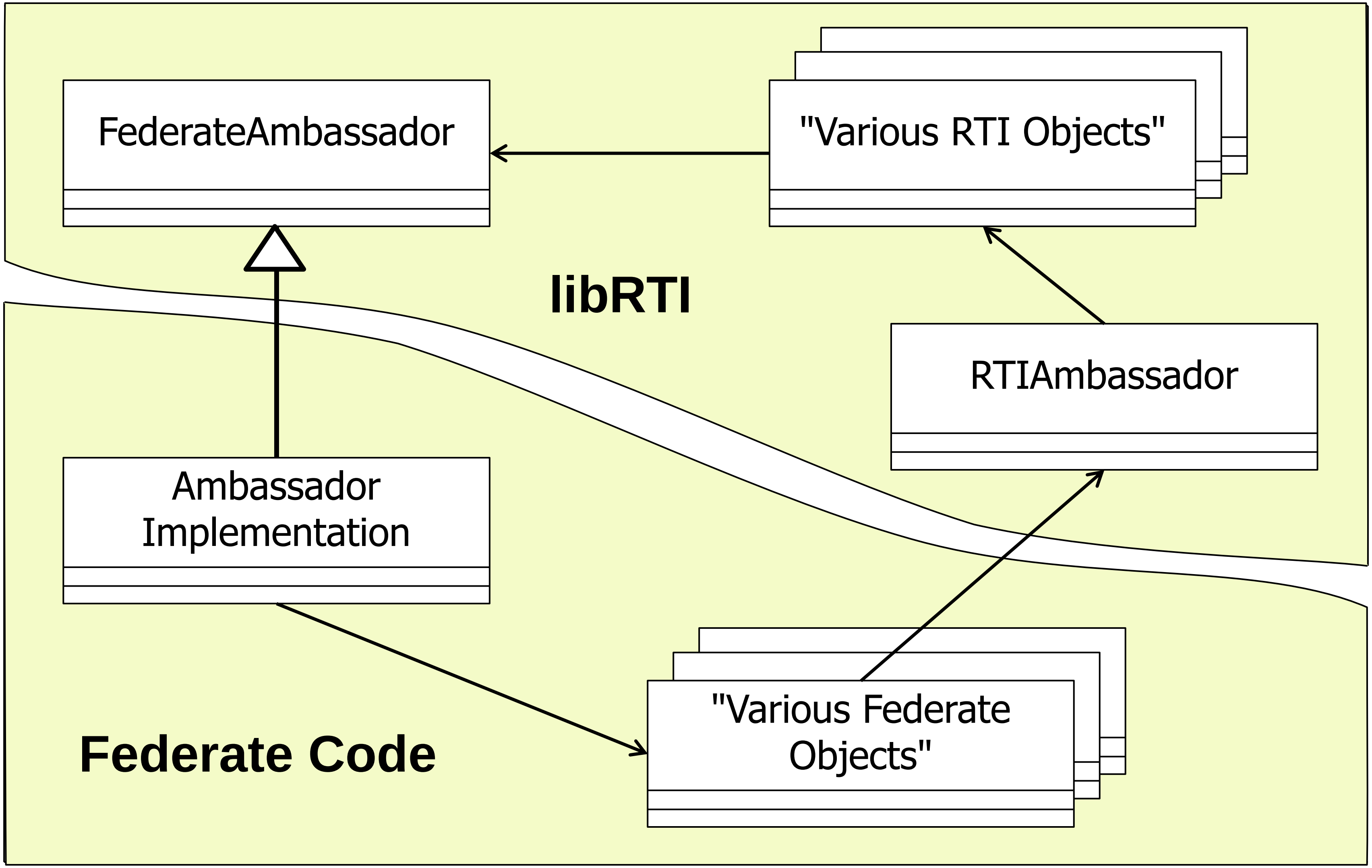
The Federate's Code provides internal functionality

The Federate's code must define the abstract RTI::FederateAmbassador class

The Local RTI Components (LRC) provide external functionality as specified by the IFSpec

The LRC include the methods for the RTI::RTIAmbassador class

RTI and Federate Ambassadors



FederateAmbassador

"Various RTI Objects"

libRTI

RTIAmbassador

Ambassador Implementation

"Various Federate Objects"

Federate Code

Sample RTI Service Request

```
try
{
 rtiAmb.timeAdvanceRequest(requestTime);
}
catch (RTI::Exception& e)
{
 cerr << "FED_HW: ERROR:" << &e << endl;
}
```

Using “Tick”

- Tick used To Wait for Service Completion
  - Allows RTI a chance to execute and respond to request
  - Waits for not more than 1.0 sec’s, nor less than .01 sec’s
- *timeAdvGrant* is a global variable, initialized to false, set *true* in callback routine *TimeAdvanceGrant*

```
timeAdvGrant = RTI::RTI_FALSE;
while (!timeAdvGrant)
{
      rtiAmb.tick(0.01, 1.0);
}
```

HLA Messages - Updates

- Send/Receive new values of attributes at end of each time-step.
  - Send: Update Attribute Values
  - Recv: Reflect Attribute Values

- Controlling unnecessary message traffic:
  - Update:
    - Enable/Disable Attribute Relevance Advisory Switch
    - Turn Updates On/Off for Object Instance
  - Reflect:
    - Enable/Disable Attribute Scope Advisory Switch
    - Attribute In/Out of Scope

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

HLA Messages - Interactions

- Send/Receive Parameters describing the event when an event occurs.
  - Send: Send Interaction
  - Recv: Receive Interaction

- Controlling unnecessary message traffic:
  - Enable / Disable Interaction Relevance Advisory Switch
  - Turn Interactions On/Off

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

Object and Interaction Registration

Objects

- Publish/Subscribe
  - *Publish Object Class*
  - *Subscribe Object Class*
- Object Registration
  - *Register Object Instance*
  - *Discover Object Instance*
- Controlling Instance Registration:
  - *Enable/Disable Class Relevance Advisory Switch*
  - *Start/Stop Registration for Object Class*

Interactions

- Publish/Subscribe
  - *Publish Interaction Class*
  - *Subscribe Interaction Class*

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

Message Order

- Two types of Message Ordering

- **TSO** (Time Stamped Order)
  - Messages delivered to federate in order of time stamp
  - RTI guarantees that no messages will be received from past

- **RO** (Receive Order)
  - Messages delivered to federate in order received

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

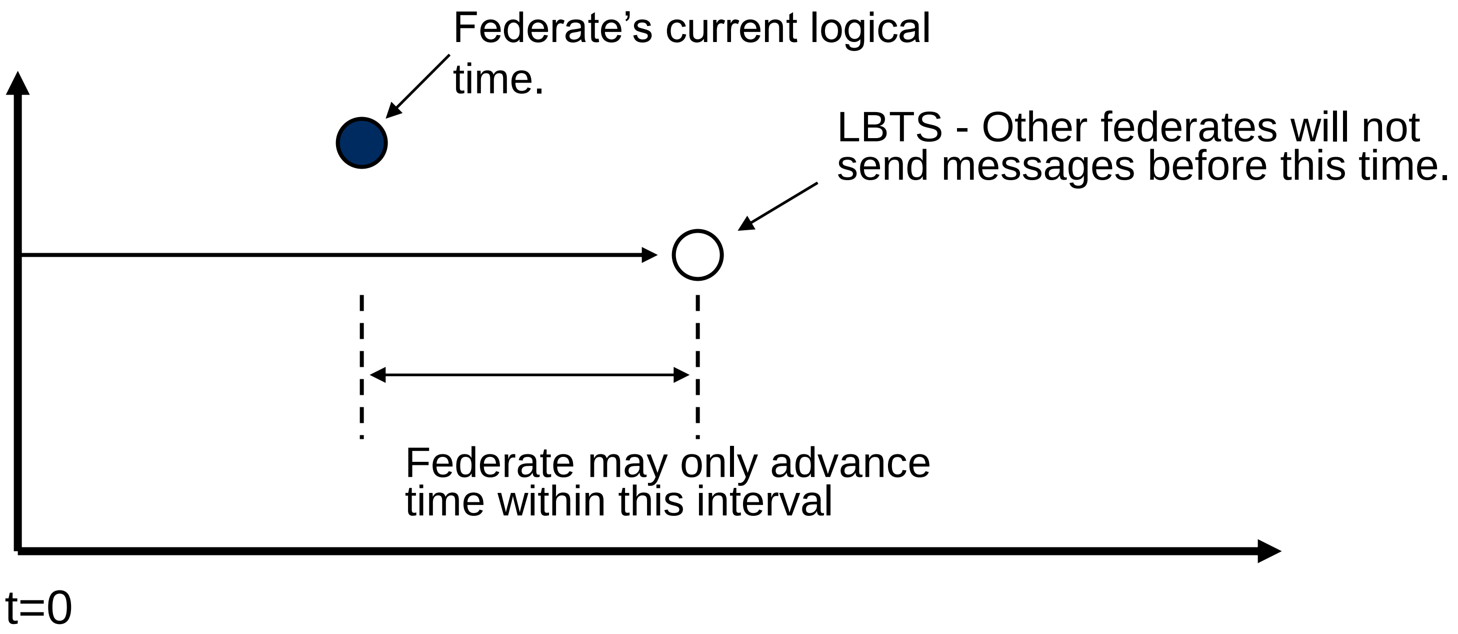
Regulating and Constrained Federates (1)

- To receive TSO messages in Time Stamped Order, Federate must declare itself **Time Constrained**.
- To send TSO messages, Federate must declare itself to be **Time Regulating**.
- By default, Federates are **neither** time constrained nor time regulating.
- To become time constrained, use RTI service *Enable Time Constrained*.
- To become time regulating, use RTI service *Enable Time Regulation*.

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

Time Constrained Federates

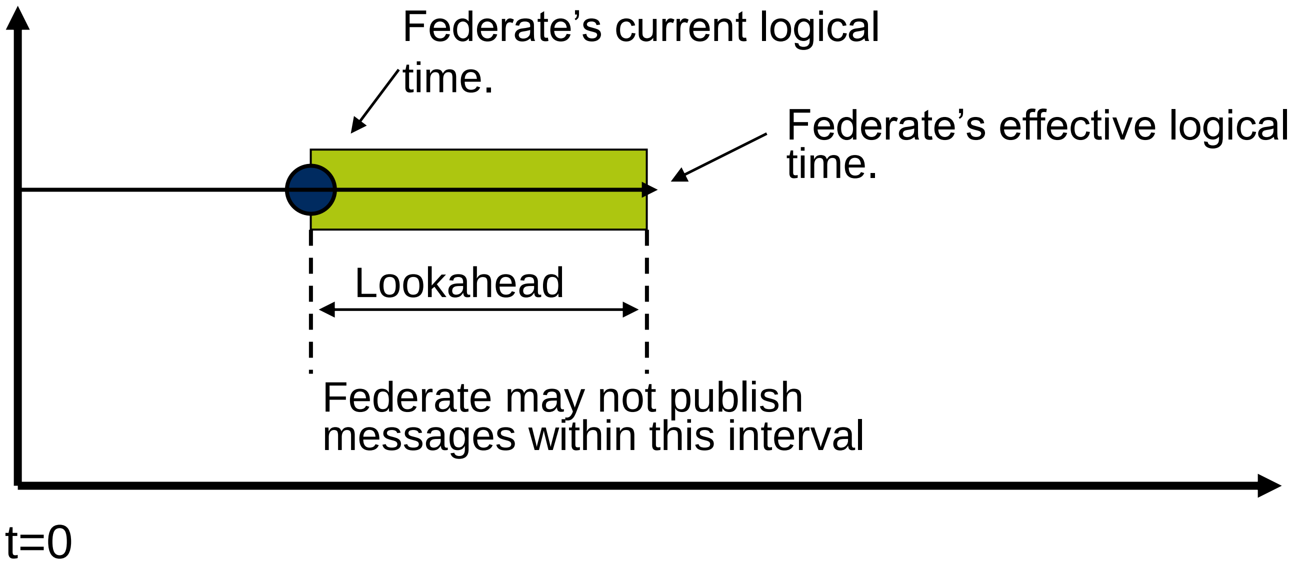
Time Constrained Federates subscribe to time stamped (TSO) data, with messages delivered in order of time-stamps.


Federate’s current logical time.
LBTS - Other federates will not send messages before this time.
Federate may only advance time within this interval
t=0

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

Time Regulating Federates

Time Regulating Federates publish time stamped (TSO) data, with messages delivered in order of time-stamps.


Federate’s current logical time.
Federate’s effective logical time.
Lookahead
Federate may not publish messages within this interval
t=0

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

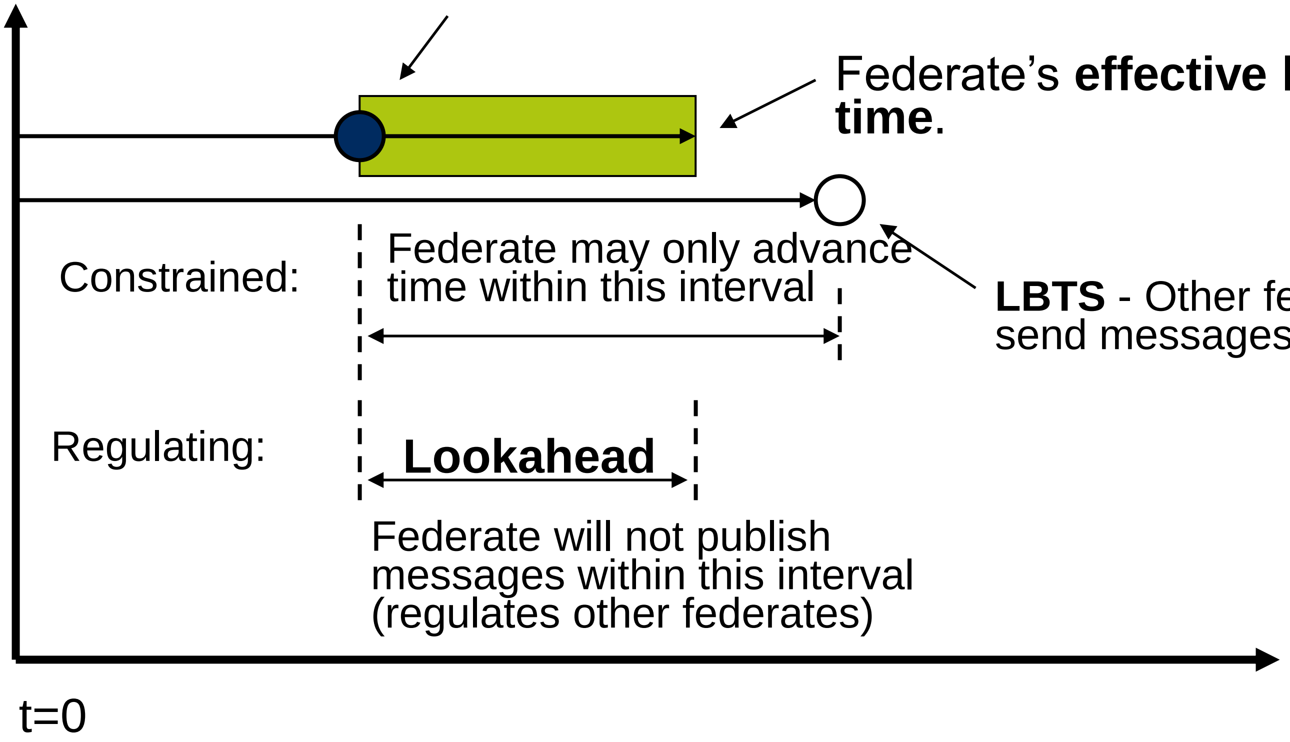
Requesting Time Advancement

- Continuous Simulations
  - To request an advance in time, use the RTI service: *Request Time Advance*.
  - RTI will notify when its ok to advance time by calling: *Time Advance Grant*

- Discrete event simulations
  - The RTI service: *Next Event Request* (t1), requests time advancement to time of next event, or to t1, whichever occurs first.
  - RTI will notify when to advance time by calling: *Time Advance Grant* , and will specify the amount of the granted time advance.

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

Federates That Are Both Time Regulating and Time Constrained



Federate's **effective logical time**.

Constrained:

Federate may only advance time within this interval

LBTS - Other federates will not send messages before this time.

Regulating:

Lookahead

Federate will not publish messages within this interval (regulates other federates)

t=0

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

Time Regulating and Time Constrained

- If a Federate sends and receives TSO data, in TSO order, it must be both *time regulating* and *time constrained*.
    - *Time constrained*: RTI prevents this federate from advancing time until it has received all messages that may be sent by other federates up to the requested time.
    - *Time Regulating*: RTI prohibits other federates from advancing time until this federate has sent all the data that it is going to send before the requested time.

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

When Are Messages Received ?

- Messages are only received when in a **time-advancing state**.

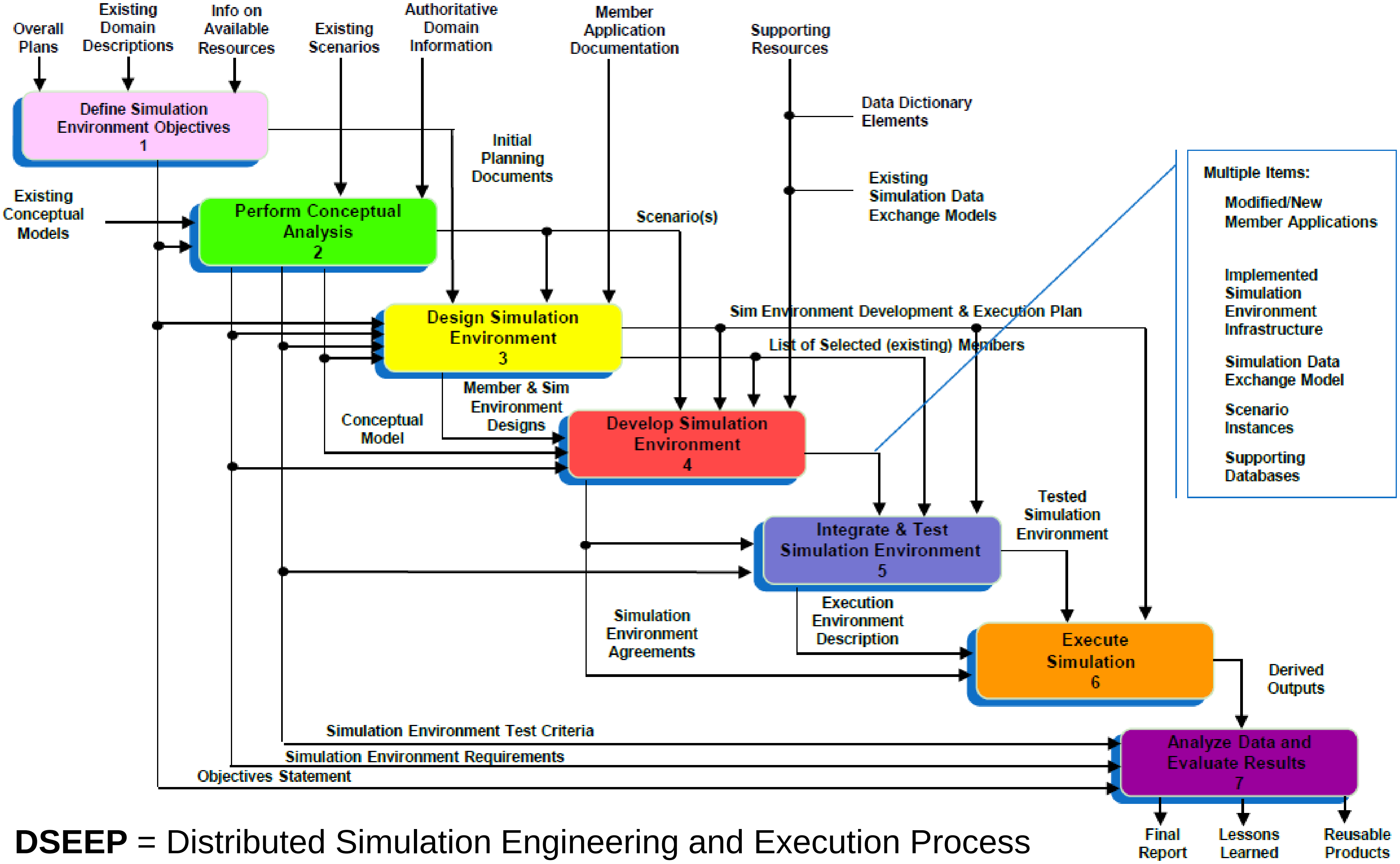
- A Federate is put into a **time-advancing state** by:
  - *Time Advance Request* OR
  - *Next Event Request*

- To enable receipt of RO messages at other times:
  - *Enable Asynchronous Delivery* (Prevents excessive delay for urgent events)

Slide based on McLeod Institute of Simulation Sciences Chico CA course on HLA

Applying HLA: DSEEP

From DSEEP / IEEE 1730-2022 presentation
by Katherine L. Morse & Robert Lutz, JHU/APL


Overall Plans
Existing Domain Descriptions
Info on Available Resources
Existing Scenarios
Authoritative Domain Information
Member Application Documentation
Supporting Resources
Define Simulation Environment Objectives 1
Data Dictionary Elements
Initial Planning Documents
Existing Simulation Data Exchange Models
Existing Conceptual Models
Perform Conceptual Analysis 2
Scenario(s)
Multiple Items:
Modified/New Member Applications
Implemented Simulation Environment Infrastructure
Simulation Data Exchange Model
Scenario Instances
Supporting Databases
Design Simulation Environment 3
Sim Environment Development & Execution Plan
List of Selected (existing) Members
Member & Sim Environment Designs
Conceptual Model
Develop Simulation Environment 4
Tested Simulation Environment
Integrate & Test Simulation Environment 5
Execution Environment Description
Simulation Environment Agreements
Execute Simulation 6
Derived Outputs
Simulation Environment Test Criteria
Simulation Environment Requirements
Objectives Statement
Analyze Data and Evaluate Results 7
Final Report
Lessons Learned
Reusable Products

DSEEP = Distributed Simulation Engineering and Execution Process

Rationale for HLA Design

- Basic premises:
  - No single, monolithic simulation can satisfy the needs of all users
  - All uses of simulations and useful ways of combining them cannot be anticipated in advance
  - Future technological capabilities and a variety of operating configurations must be accommodated
- Consequence: Need **composable** approach to constructing simulation federations

Critique on HLA

- Military, no breakthrough in the civil world
- Overkill for simple applications
- Simulation is not monolithic anymore, but replaced by a centralized, monolithic RTI
- DSEEP process only works in a command-and-control setting