Topologies for integrating System

Connection between federation system made by :

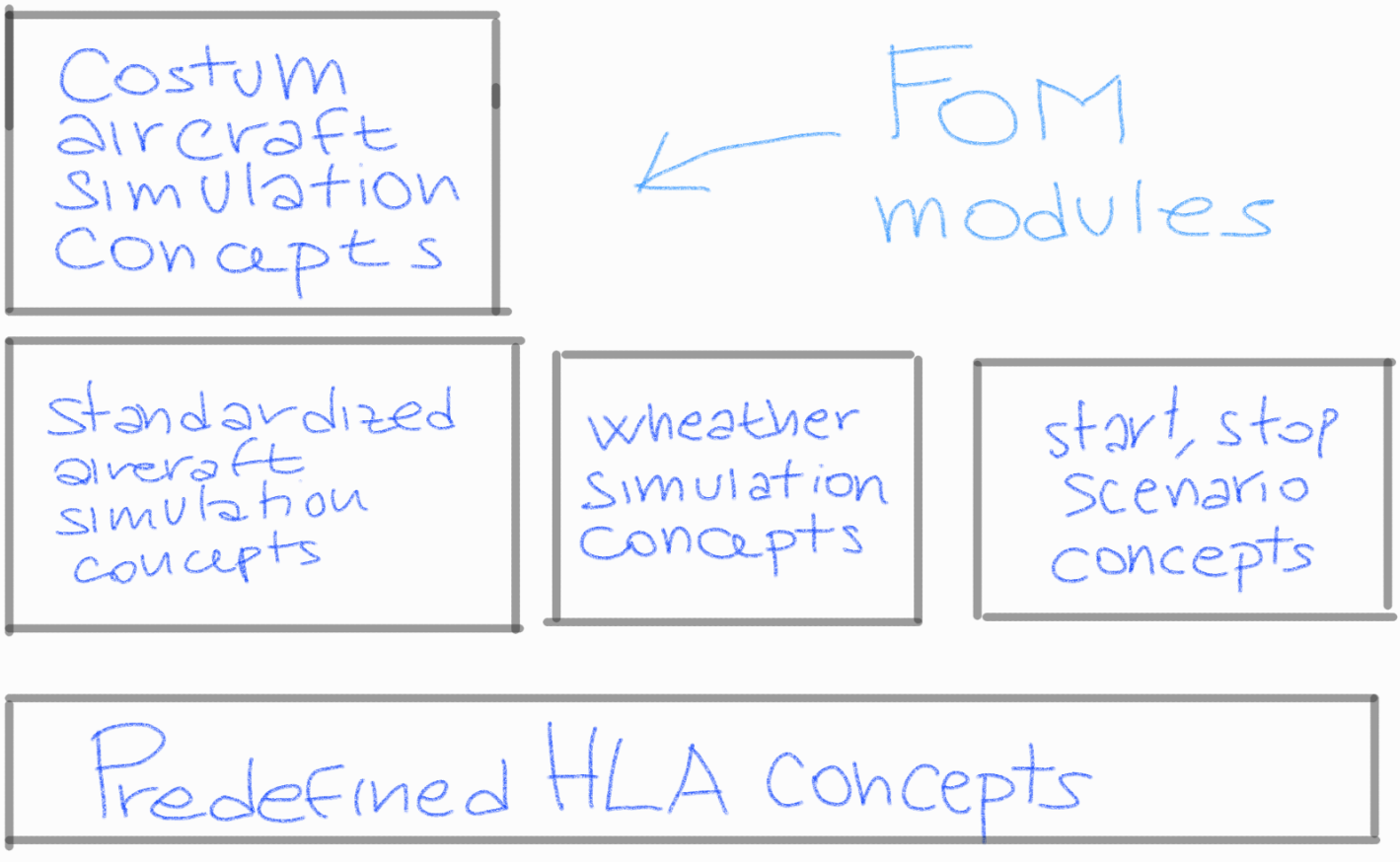
- Pair wise integration --> works well with few systems because it established an agreement for each pair of systems to what kind of services exchange , that's make clear the exchange. However, due to the fact every systems has to knows pairwise system' s characteristics when a new system is added the number of connection increase and a lot of existing systems has to be modified.
- Service bus (used in HLA) --> each system provides or consumes a particular services that is interested in. Each system can be replaced by another system without updating several other systems. Nwe systems that produce or consume services can easily be introduced. This is more FLEXIBLE and SCALABLE.

FIVE IMPORTANT CONCEPT :

- RTI --> software part, provides the HLA services. One of them is to send the right data to the right receiver.
- Federates: systems connected to RTI, typically a Simulator.
- Federation: all federates + RTI + FOM.
- Federation object model: document of description of data exchange in the federation (objects and interactions).
- Federation execution: session when Federation runs, for example a pilot training session when you run several flight simulators. If you run the federation several times you will have several federation executions.

FOM CONTENT

object classes, interaction classes and Data types.
Example:
Object--> car (something that persist over the time)
Interaction --> start (something that don't persist over the time)
Datatype--> nameString, positionRecord, SpeedInteger (describes semantics and technical representation).
Fom can be extended over time without breaking existing simulations.
FOM follows a format called the HLA Object Model Template, which is based on XML, and follows HLA standard.
There are also XML Schemas that can be used to verify the format of an object model.



HLA SERVICES

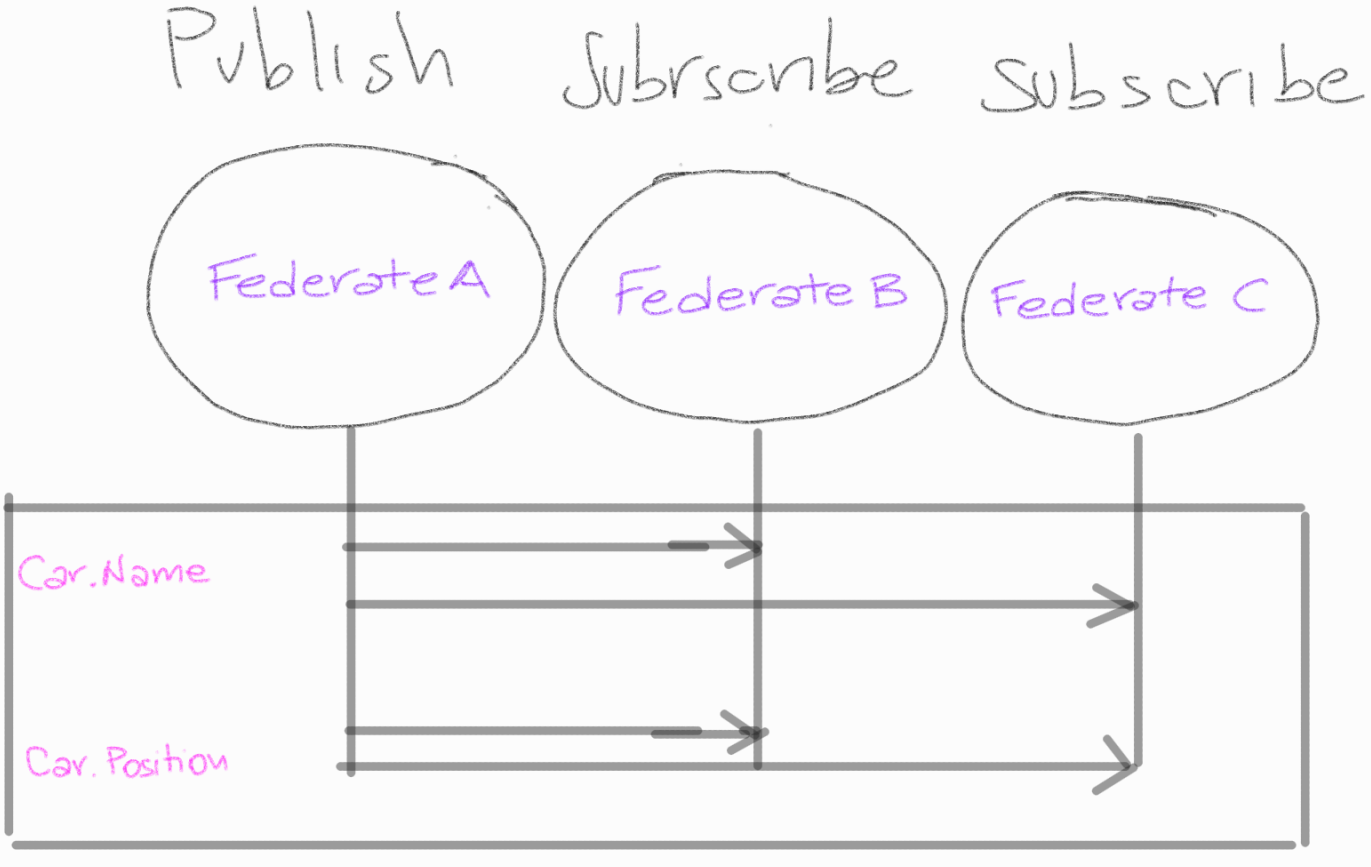
Services are implemented by RTI. Divisible in 7 groups.
Categorizziamo questi gruppi in:

- Information exchange services
- Synchronization services
- Coordination services

To use these services a federate will make calls to the RTI and receive callbacks from the RTI, which means that there is two way communication.

Information services

A Publish/subscribe scheme enable federates to exchange data. RTI keeps track of it and delivers data from publishers to subscribers in a matrix form.



Syncronization services

3 types of synchronization service.

- Handling og Logical time. Simulation run both in real time or in fake time, to be able to correctly exchange data with time stamps, the RTI coordinate both how fast the simulators advance logical or scenario time as well as correct delivery of time stamped data.
- Syncronization points that enables members of the federation to coordinate when they have reached a certain state, for example when they are ready to start simulating the next phase of a scenario.
- Save/restore thar makes it possible to save snapshot of a simulation at a given time. Useful to rerun a scenario with different parameters and for backup purposes.

Coordination services:

- Management of FedExec and federates that are joined to a FedExec
- Transfer of modelling responsibilities.
- Advanced inspection and management of the federation. --> described in MOM

Service grouping in the HLA standard

1. Federation management--> keep track of Federation executions and federates syncronization points (save/store).
2. Declaration management --> publish and subscribe of object and interaction
3. Object management --> registering and discovering object instance, updating and reflecting attributes, sending and receiving interactions.

4. Ownership management --> transfer of modeling responsibilities
5. Time management --> handling of logical time (time stamped data and advancing federate time).
6. Data Distribution management --> filtering based on data values
7. Support services --> utility functions
8. Management object model --> inspection and management of the federation

HLA as A STANDARD

Designed to facilitate interoperability among simulations and to promote reuse of simulation and their components. The HLA is composed of three major :
1.HLA rules : set of ten basic rules that together describe the general principles defining HLA.
2. HLA INTERFACES: description of interface between federates and RTI.
3. HLA object model template: a specification of the common rmat and structure for documenting HLA object models.

FEDERATION RULES of HLA

10 rules : 5 for federete and 5 for Federation that tdefine how to deal to carry out interoperability. Describes the simulation and federate responsibility.

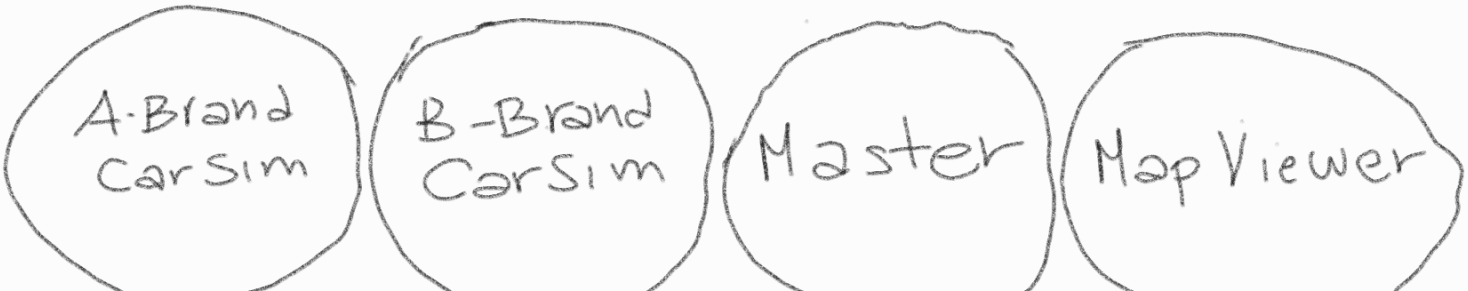
5 RULES OF FEDERATION:
1. Each Federation shall have a federation object model, to define how it organised, documented in accordance with OMT.

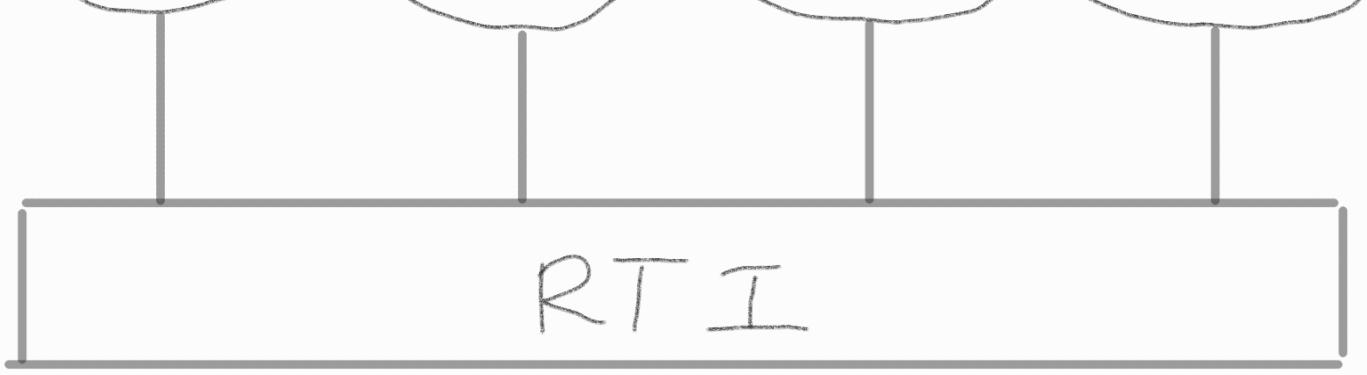
2. All rappresentation of objects in the FOM shall be I the federates, not in RTI. THAT MEANS RTI DOESN'T INCLUDE ANY ELEMENT OF FEDERATION.
3. During a FedExec , all exchange data among federates shall occur via the RTI.
4. During a Federation execution, federates shall interact with the RTI I accordance with the HLA interface specification
5. During FedExec an attribute of an instance of an object shall be owned by only one federate at any given time

5 RULES OF FEDERATE:
6. Every federates shall have a SOM (SIMULATION OBJECT MODEL), documented in accordance with OMT
7. Should be able to update or reflect any attributes of objects in their SOM
8. Federates should be able to transfer or accept ownership of attributes dynamically during a FedExec
9. Federates should be able to vary the conditions under which they provide updates of attributes of objects
10. federates should be able to manage local time in a way which will allow them to coordinate data exchange with other members of a Federation.

Connecting to a federation

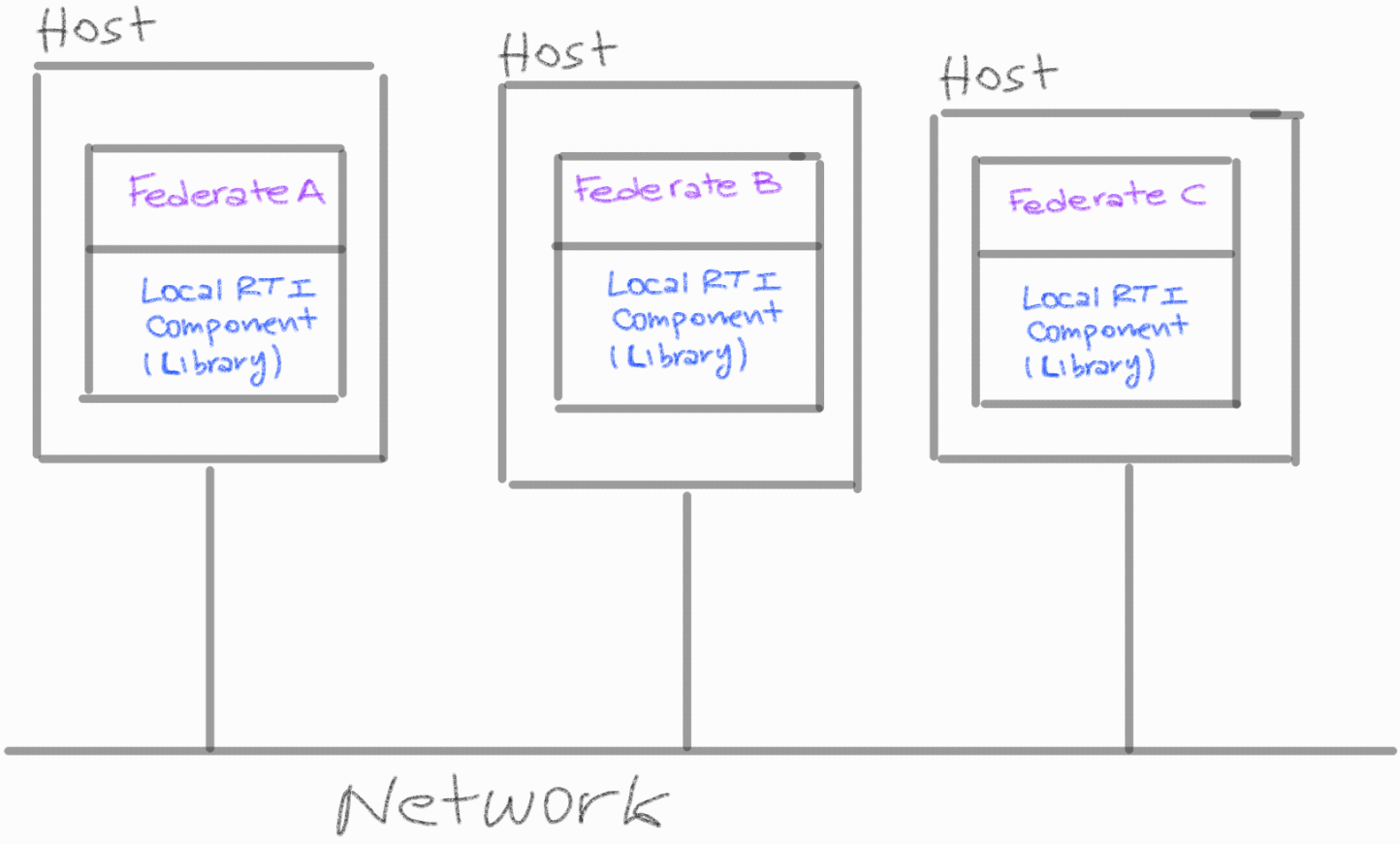
Example application:
Fuel Economy Federation --> evaluate how far cars from different manufactures can drive using a limited amount of fuel. Here a diagram of the federation :





Master is the management federate from which an operator can set up the scenario and start and stop simulation.
Map Viewer is a federate that displays a map with cars and their fuel status.

The central and local RTI components



Local RTI Component --> local library installed on each PC with a federate.
Central RTI Component --> application that coordinates the FedExec and keeps track of a joined federates. Needs to be started on a address pc and before any federate can join a Federation.

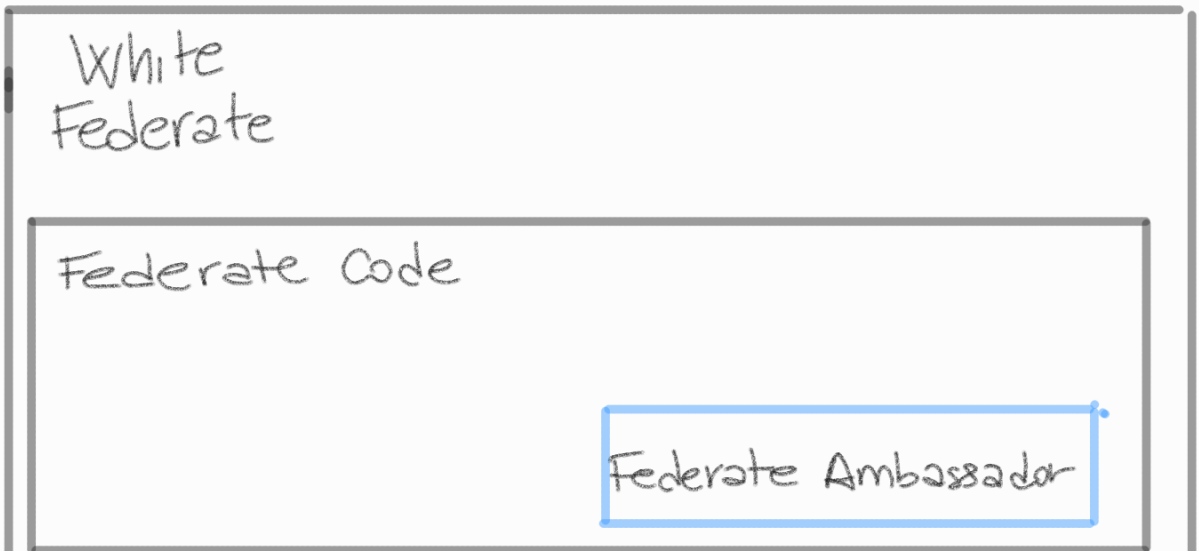
RTI

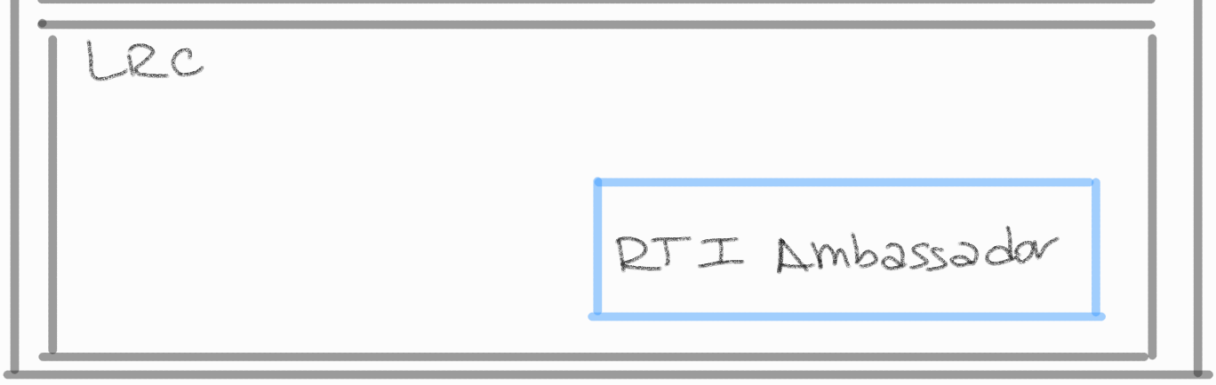
Software that provides common services to simulation systems. Implementation of the interface spec. An architectural foundation encouraging portability and interoperability. Separate simulation & communication. Improves on older standards. Facilities construction and destruction of federations. Supports object declaration and management between federates. Assists with federation time management.

How to compound a federate

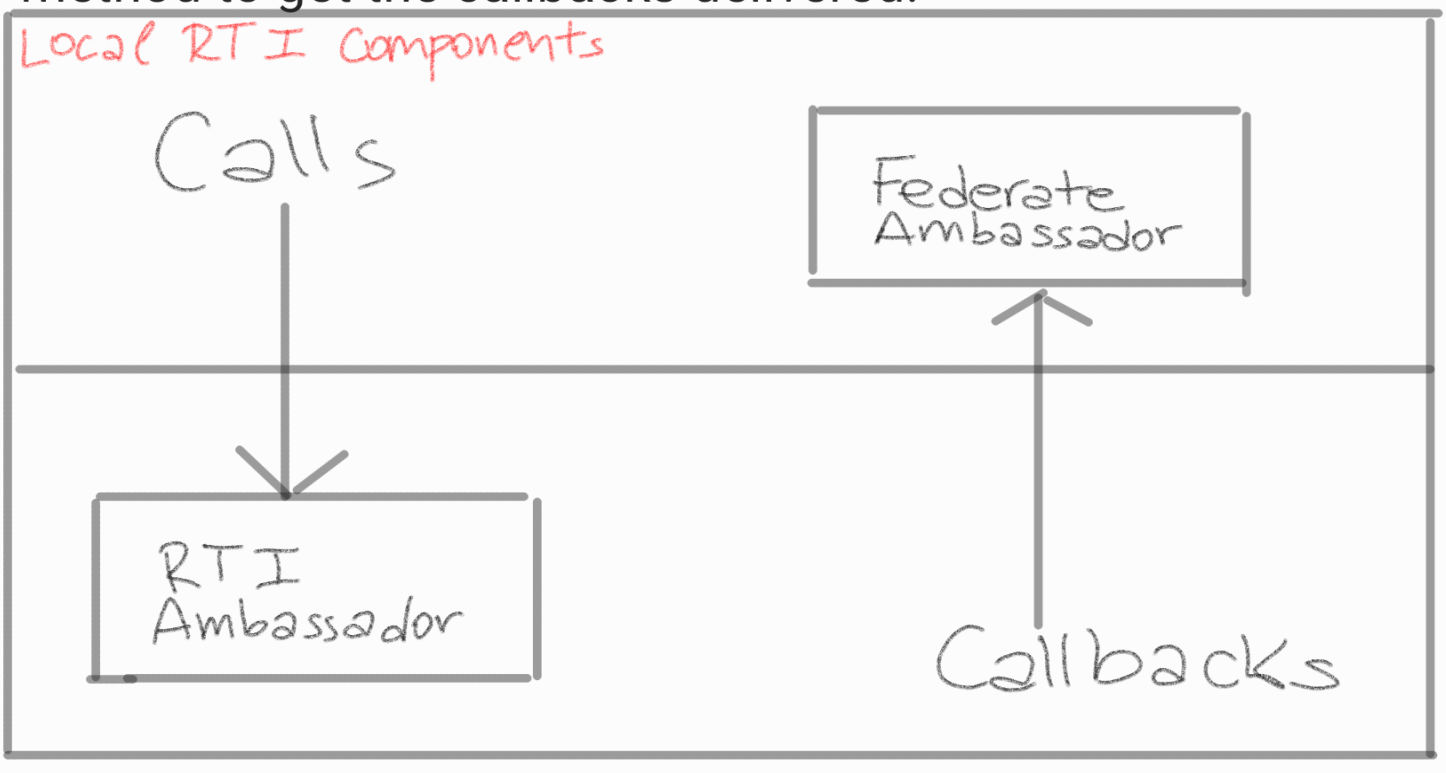
FEDEXEC: goal is to allow user to join federation. Assign an identifiers to each attributes of an instance or to an instance of an object. When subscribe to an objects in reality give an attributes go handle the object. Facilities exchange among federates.

RTIEXEC: manage multiple federations creation and destruction. In this network we can run a Federation but also run another federation on a different simulation, or another federation on a different machine but same name of the first (like open 2 program on the same page, PowerPoint and word for example).
Global Process run on unique platform.





2 callbacks model : IMMEDIATE and EVOKED. The first allow RTI deliver the callbacks in a separate threads as soon as they are received. The latter, I have to call explicitly the method to get the callbacks delivered.



HLA services detailed

- Connect: service that connect a simulation to an RTI. Parameters are provided by a configuration file called Local Settings Designator. Need to specify type of callbacks. An issue is the connection Failed exception, which occurs when there is on RTI available given the specified network address.
- Create federation execution: a FedExec is created with a specific name. Need to provide a list of valid FOM modules and be sure to not create an exception for not

be able to locate the FOM Module as well as invalid FOM modules.
- Join FedExec: makes the simulation a member of a FedExec. Needs to have the name of fedexec to join; needs to handle the exception when the federate name is already used.
- Resign FedExec: resign the federate from the federation.
- Destroy FedExec: it's possible every time a resign happen but only if there is no federate in the federation already.
- Disconnet: disconnect the federate from the RTI.

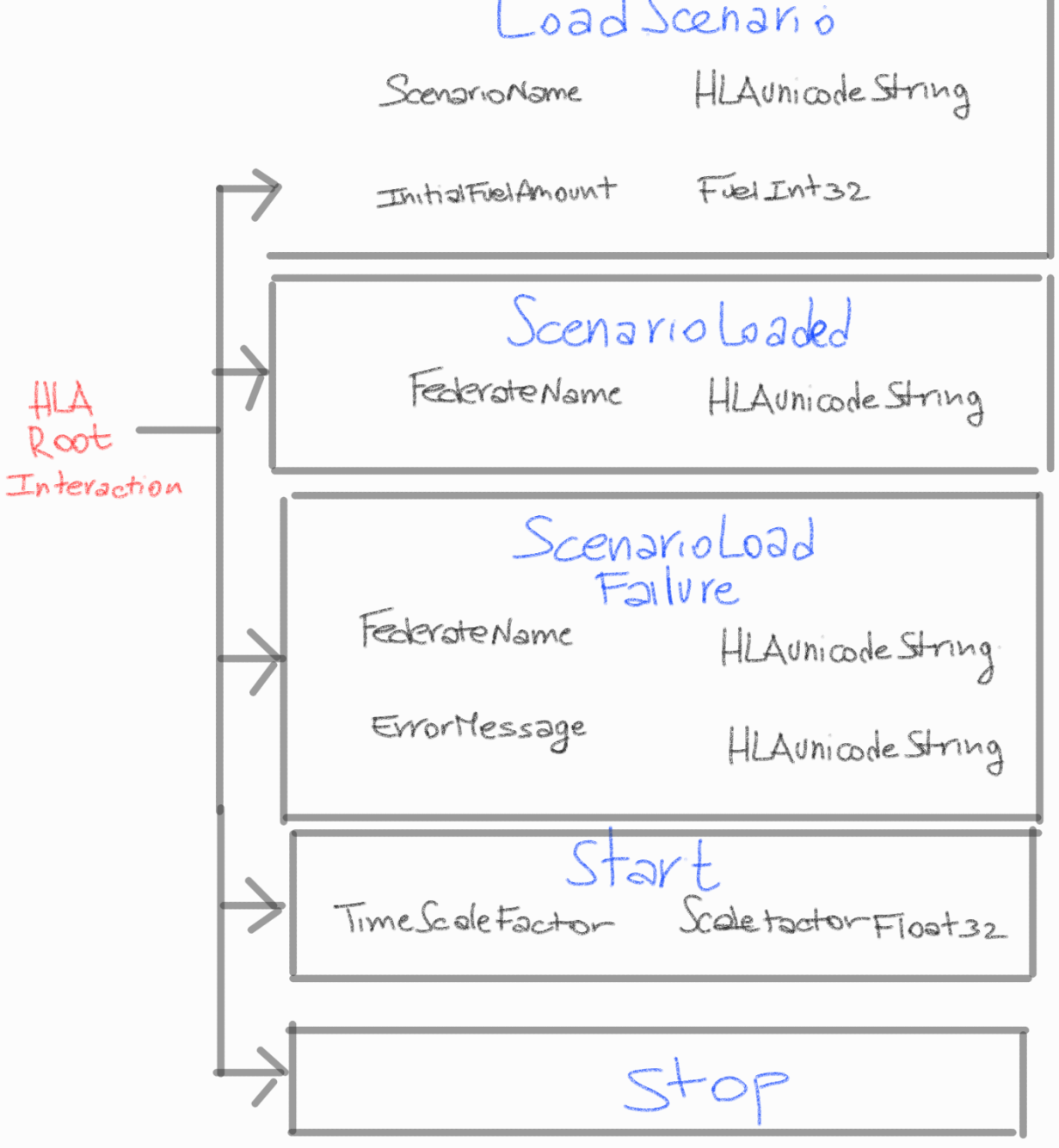
INTERACTIONS - CREATING THE FOM

FOM is implanted in XML form but it can be displayed as tables. In this case the FOM has to be visualized in FOM editing program Pitch Visual OMT.
The following items are the most relevant:

- Identification table
- Interactions with parameters
- Data types

Identification table:
Fill it with name, purpose, date and version and info about me. Apply configuration control of your FOM for example in a version tracking system.

Interactions with parameters:
7Definition of Interaction classes--> are subclasses of the predefined class HLAinteractionRoot



Loads scenario is used to inform all participating federates about scenario to run. Interaction is above.
The interaction will be both Published and Subscribed to by federates upon the federations. There are two parameters : scenarioName , which is used to predefined data type HLAunicodeString; initialFuelAmount, which uses a user defined data type called FuelInt32. --> we use a Simple data type FORMAT for simple data of attributes of fuel.

A Simple data type:
Alcuni valori da settare nel dialog form del simple data type FeulInt32:

- **Representation** --> exact representation of data when they are exanchanged between federates.
- **Unit** --> for measuring fuel is the metrical liter
- **Resolution** --> of the value is one liter
- **Accuracy** --> is one liter

These parameters are extremly important to avoid errors during the transmissions of the data.

More about FOMs:
FOM Module --> numbers of related objects classes, data types that are sorted in a separate file. Modular FOMs simplify both the development and maintenence of FOMs as well as their use.
MIM --> special module called management and initialization Module. It contains all of the predefined concepts of HLA. This module is provided automatically by the RTI when a federate calls the create FedExec servic.
Classes and Datatype in MIM starts with HLA.

INTERACTIONS - CALLS FOR SENDING AND RECEIVING

Initial preparation for interactions:
1. Allocate helper objects for correctly encoding and decoding data according to the FOM. We create them before to better performance.
2. Get handles, which is a type of reference, for the Interaction class and its parameters. Handles are used in RTI calls
3. Publish and subscribe to the interaction class. These are important to because they tell to RTI about which federates send datatype and which receive datatype.

Encoding and deciding helpers:

Not RTI services?, they are utilities.

HLA services detailed of this class:

- Get interaction class handle: this service returns a handle for the specified interaction class in the FOM.
- Get parameter handle: returns a handle for the specified parameter of an interaction. The interaction class handles need to needs to be supplied.
- Publish interaction class: informs the RTI that the federate publishes the specified interaction, which means that it can send such interaction.
- Subscribe interaction class: informs the RTI that the federate subscribers to the specified interaction, which means that the federate will be notified whenever another federate sends such an interaction.
- Send interaction: this service sends an interaction and the supplied parameters values. To send an interaction, we will encode the parameter values and send the interaction. When we send the interaction we can also provide a user supplied tag. In this case it is empty.
- Receive interaction (callback): 3 versions of the RecceiveInteraction method for FederateAmbassador in the APIs. Are called depending on hoe many optional parameters are presents.

Receiving interactions

The RTI and federate interactions are the callback. Responsibility of federate code: federate code implement/ run the ambassador --> create the Federation Object Variation, that call RTI ambassador --> that Update RTI Object Variation --> RTI have all the object inside and guarantee

federate ambassador functionality.

Example of FedExec Analysis (workflow diagram):

PROCESS STEPS

User start RTI --> RTIExec --> user became federated--> federated start federation --> FedExec --> other federates join federation in FedExec phase

If there is not a syncronization timer, when other federate join their are not syncronized refer to a start point...but we don' t care.

FEDEXEC ANALYSIS

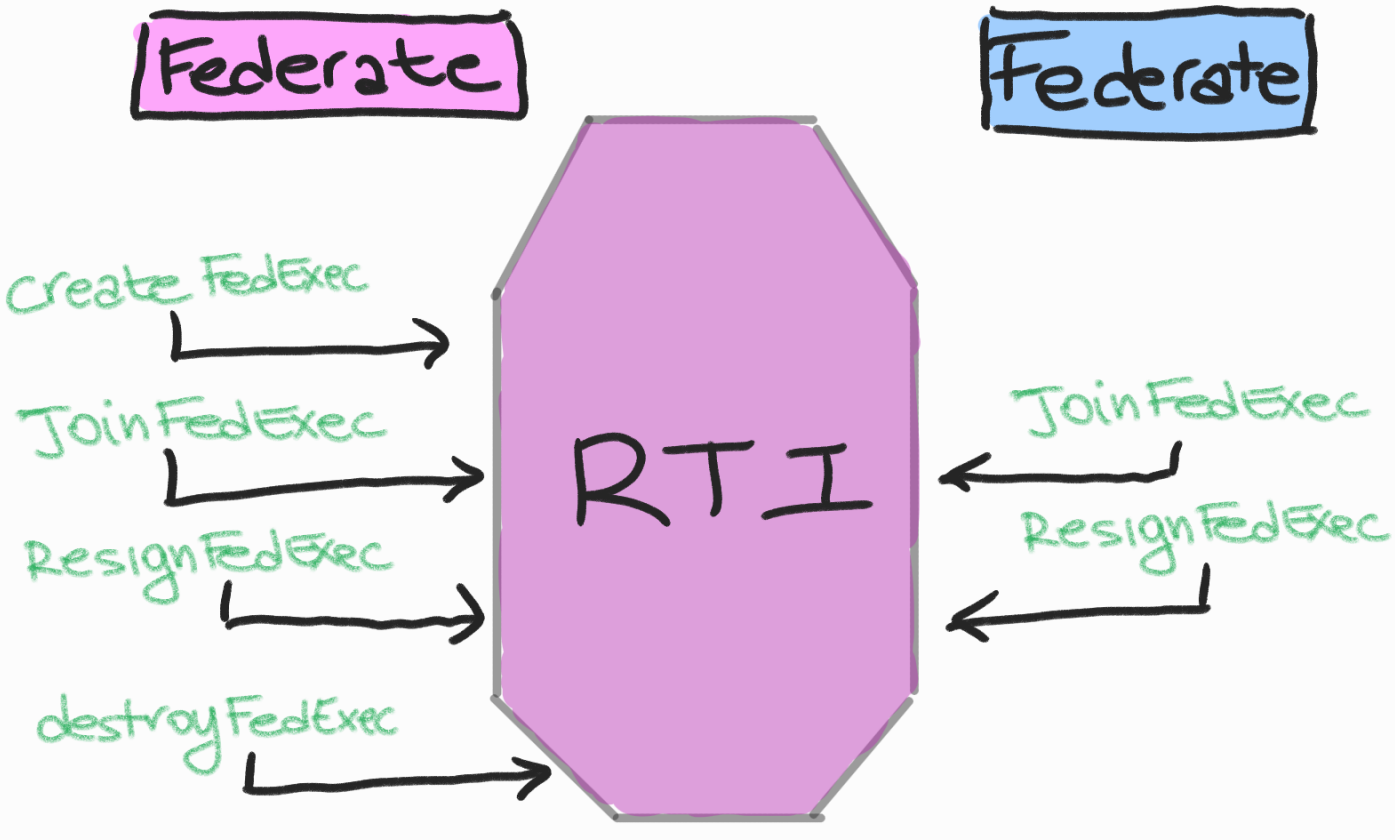
Base flow:
Federation creation --> statements manage--> management of temporal statements --> support services --> objects management--> time management --> Federation destruction

Exeception: time is expired, time is not valid, timeadvanceRequest still pending, federate is not exec member, rescue in progress, restore in progress, RTI internal error

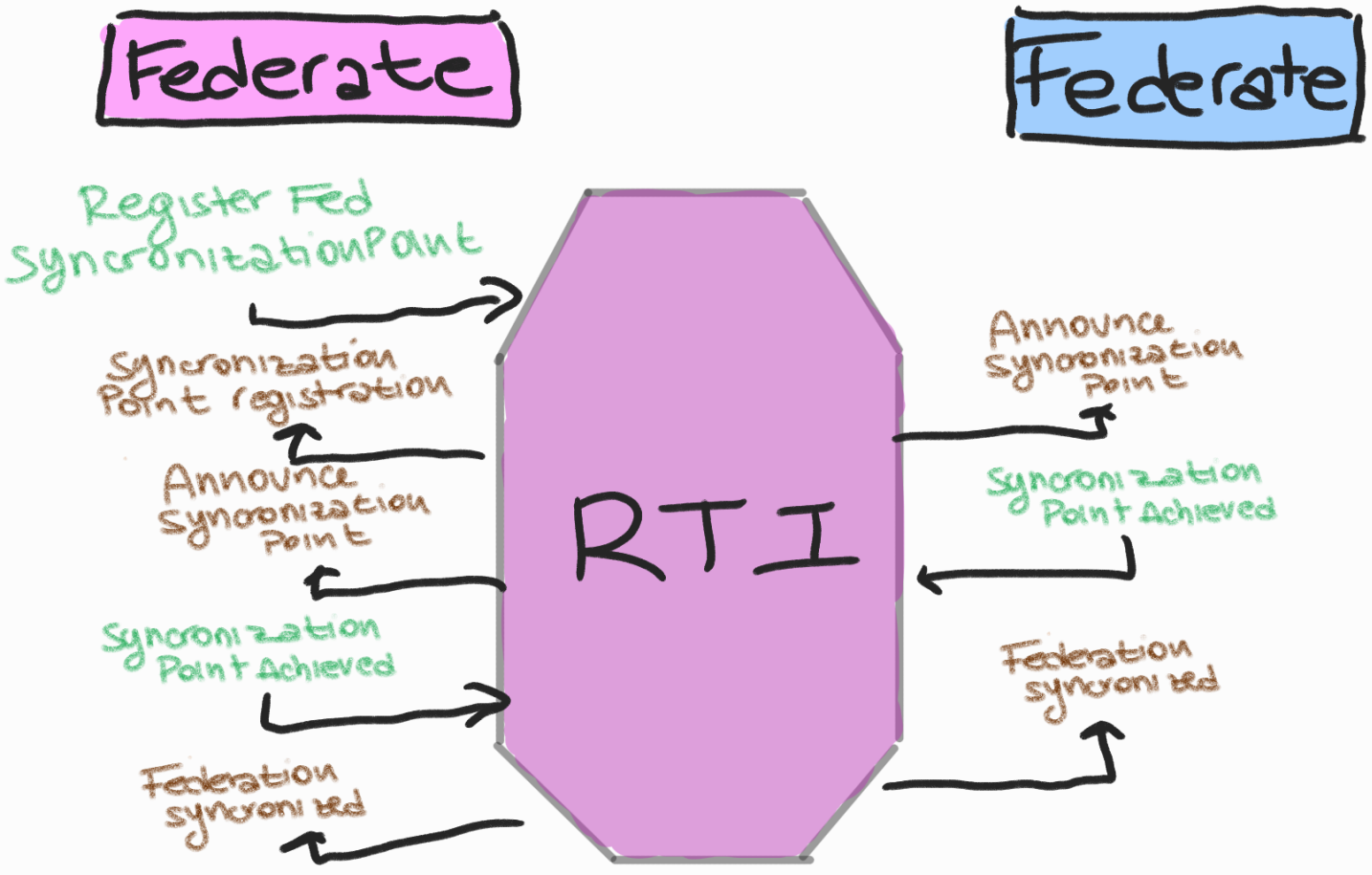
The "Tick": some kind of refresh frequency that define how often the federation ambassador check some of the situation. By changing tick I can increase control loop and same time I can speed up simulation if I have latency issues.

FRDERATION SERVICES MANAGEMENT

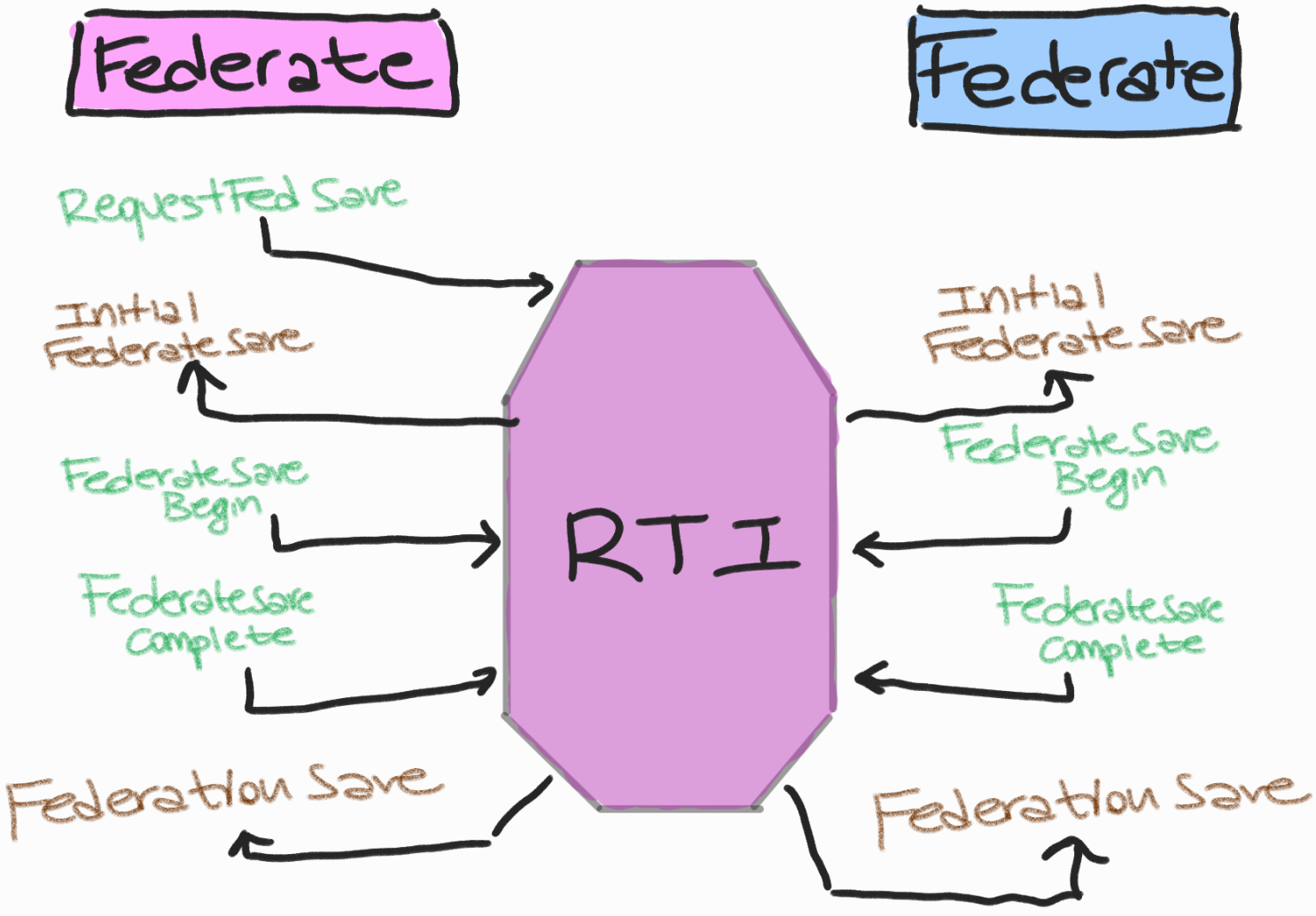
Life cycle management



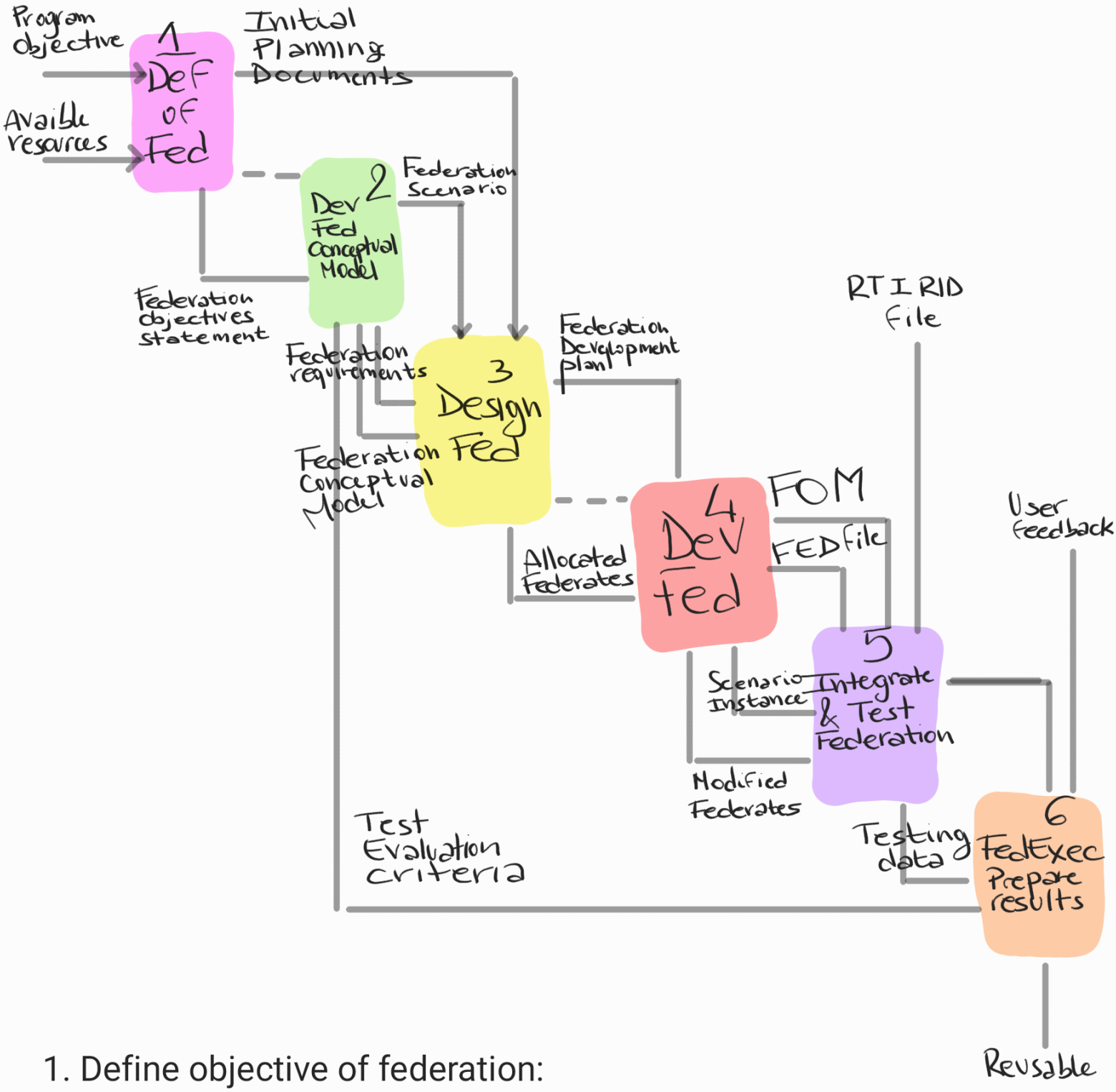
Syncronization management



Save management



FEDERATION DESIGN PROCESS - FEDEP



1. Define objective of federation:
Identify user and sponsor nees, develop objectives.
Resource defined could be: software language, compiler, tools etc

2. Develop federation/ conceptual model
Federation requirement so the input of design, the conceptual model, the SCENARIO. This last is very important because allow you to specify what you have to do.

3. Design Federation
Select federates

Developed what kind of Federation design.
Prepare a plan.

4. Federation development
Output the FED file, the scenario instance and modified federates.
Develop FED --> Federation Execution Data
Develop FOM --> Federation Object Model
Establish Federation agreements
Implement Federation designs
Implement Federation infrastructure

5. Integrate and Test Federation
First test is to see if federates and federations join.
Input: RTI RID file --> RTI INTERCHANGE DATA file
Output_1: tested Federation
Output_2: tested data

6. Execute Federation and prepare Results
Take in input the outputs of 5 and the test evaluation criteria from 2. Outputs : user feedback and reusable

OMT - object model template

Provides a common framework for HLA. One of the goal is reusing different components.
We have informations such as Objrct Class structure table, Object Interation table, Attribute /parameter table, FOM/SOM lexicon.

FOM --> one per Federation. it define all info's that are shared and refer to all the issues of interfederate relationship.

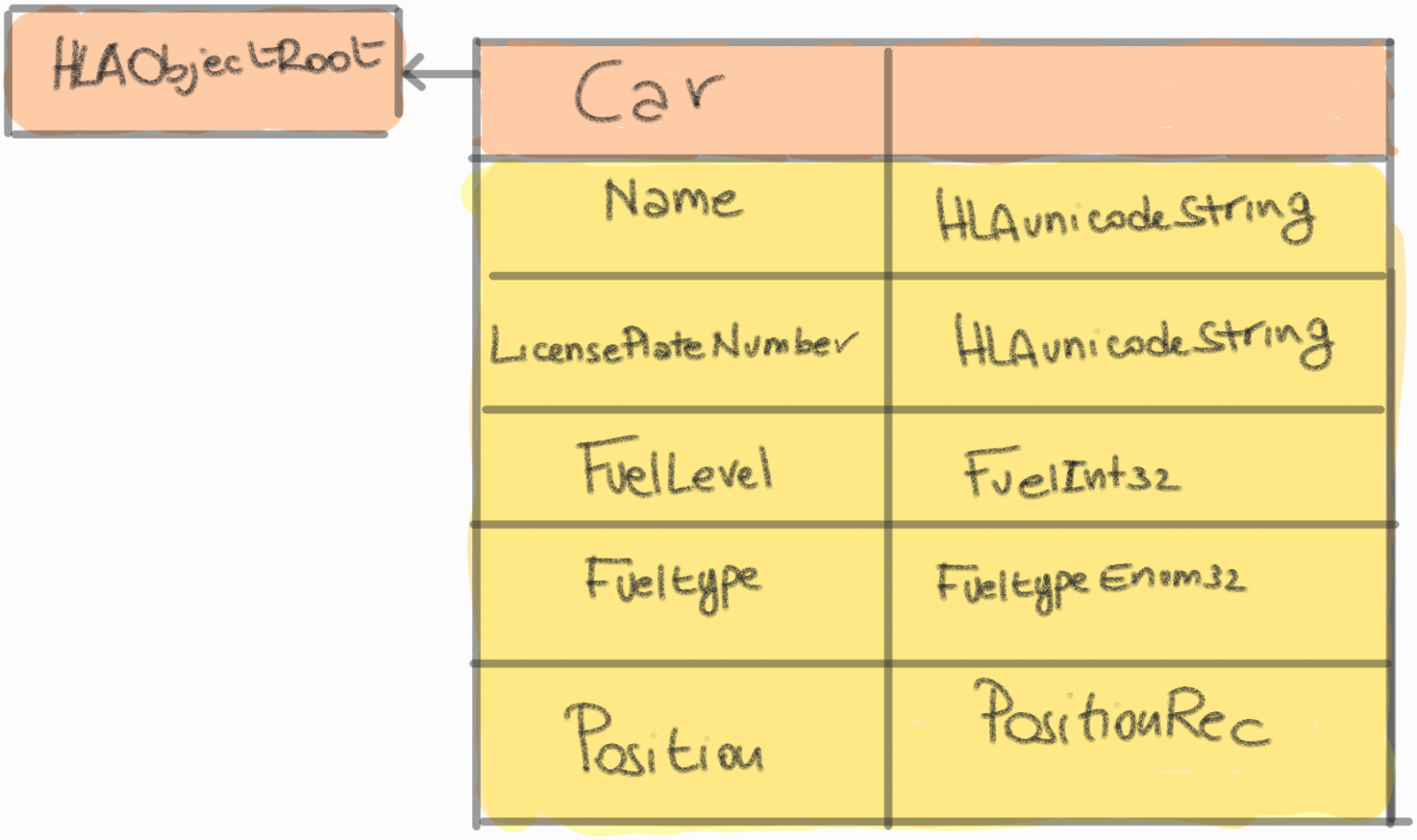
SOM --> one per federate. Describe characteristics of federate,describes inside operations.

MOM --> unibpversal definition. Identifies objects and interactions

used to manage a Federation.

Objects - creating the FOM

Object class - The Car



This class is a subclass of the predefined HLAobjectRoot and can both Published and Subscribed in the federation. Every attributes of this subclasses can be updated in the apposite dialog form in the section "Update".

Objects - the management

STEPS:

- Initial preparation
- Registration of an object instance
- Discover object instances
- Reserving a name for instance

In this phases HLA has some services like:

- Get object class handle --> returns a handle for a specified object class in the FOM.
- Get attribute handle --> returns a handle for the specified attributes of an object. The object class handle needs to be supplied.
- Publish objects class attributes --> informs the RTI that the federate publishes the specified object class / that it can send updates for object attribute so that means it can register object instances.
- Subscribe objects class attributes --> informs RTI that federates subscribes to a specified object class / that wishes to receive updates for attributes and so that means that it wishes to receive notifications when a new object is registered by another federate.
- Register object instance : unique name and unique handle is created for the new instance registered.
- Discover object instance --> callbacks

MANAGEMENT OF OBJECTS DECLARTIONS:
Definition what data send and what receive.
Object management refresh and updates--> RTI receive interaction and so the parameters to refresh.
Object class: attributes of federate that are maintained in the time. Venn diagram to define and visualize what object you want to describe.
P&S of objects: one federates publish attributes another subscribes federates --> all visualized with Venn diagram.
P&S interaction refresh and updates like object management. Interactions are produced as "all or nothing" and it's not possible specify which parameters will be publish in an interaction.

Refresh of an object: to create an object the federate must before publish its class. The federate registers the new instance of the object. To discover this object, other federates must have subscribed to that object class. Those federates then will discover the object instance.

Updates for an attribute:

- Static: value is assigned and Update at the beginning of the simulation and doesn't change.
- Conditional: updates is sent when a new value arrive from "on change" condition.
- Periodic: the value is sent out periodically even if it didn't change.

When updates:

- When the class has a new value
- When another federate request a value.

Service update : it sends an attribute update for a particular object instance.
Provide attributes value update --> callback
Reflecting attributes values --> callback

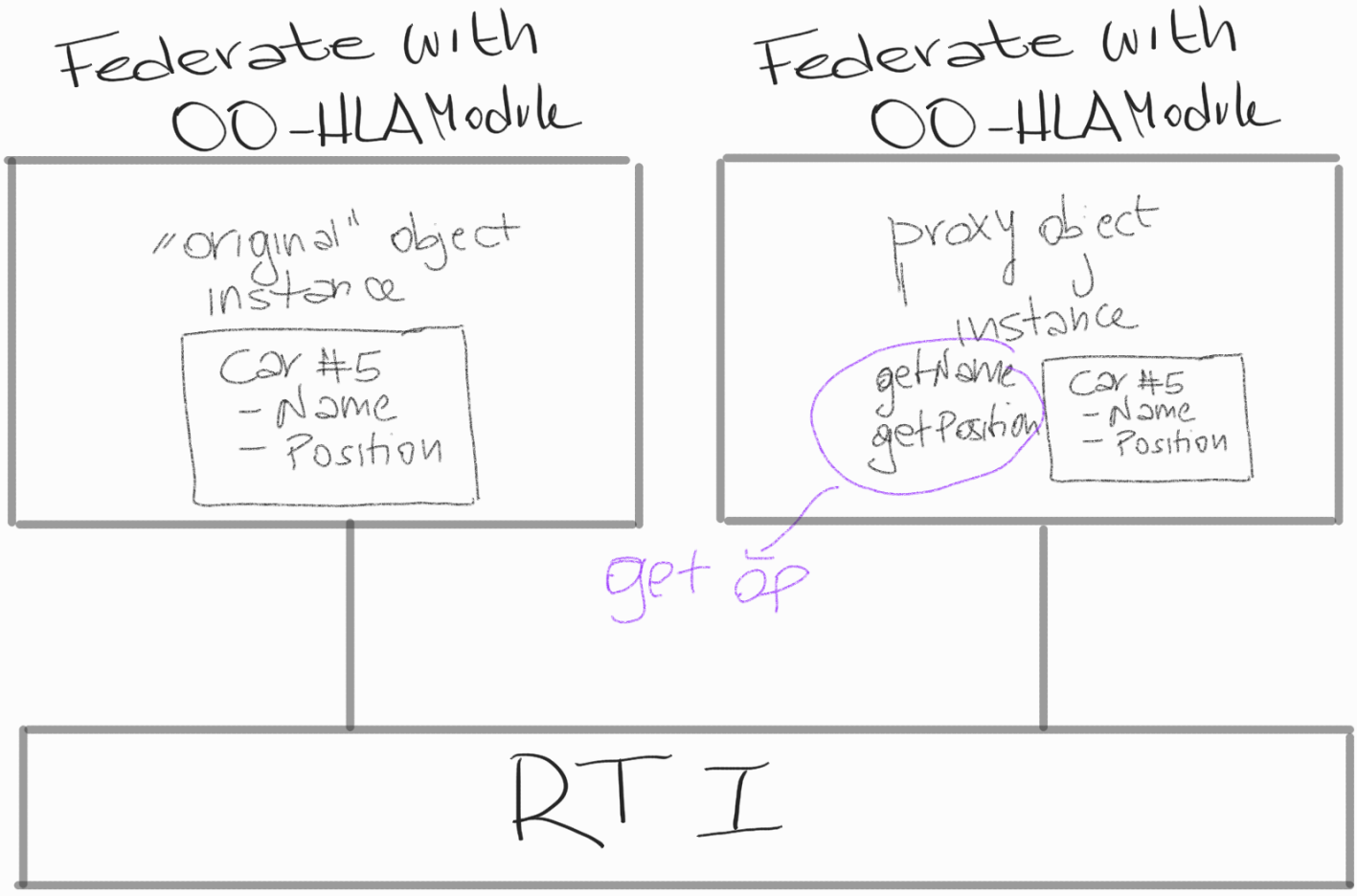
Object oriented HLA

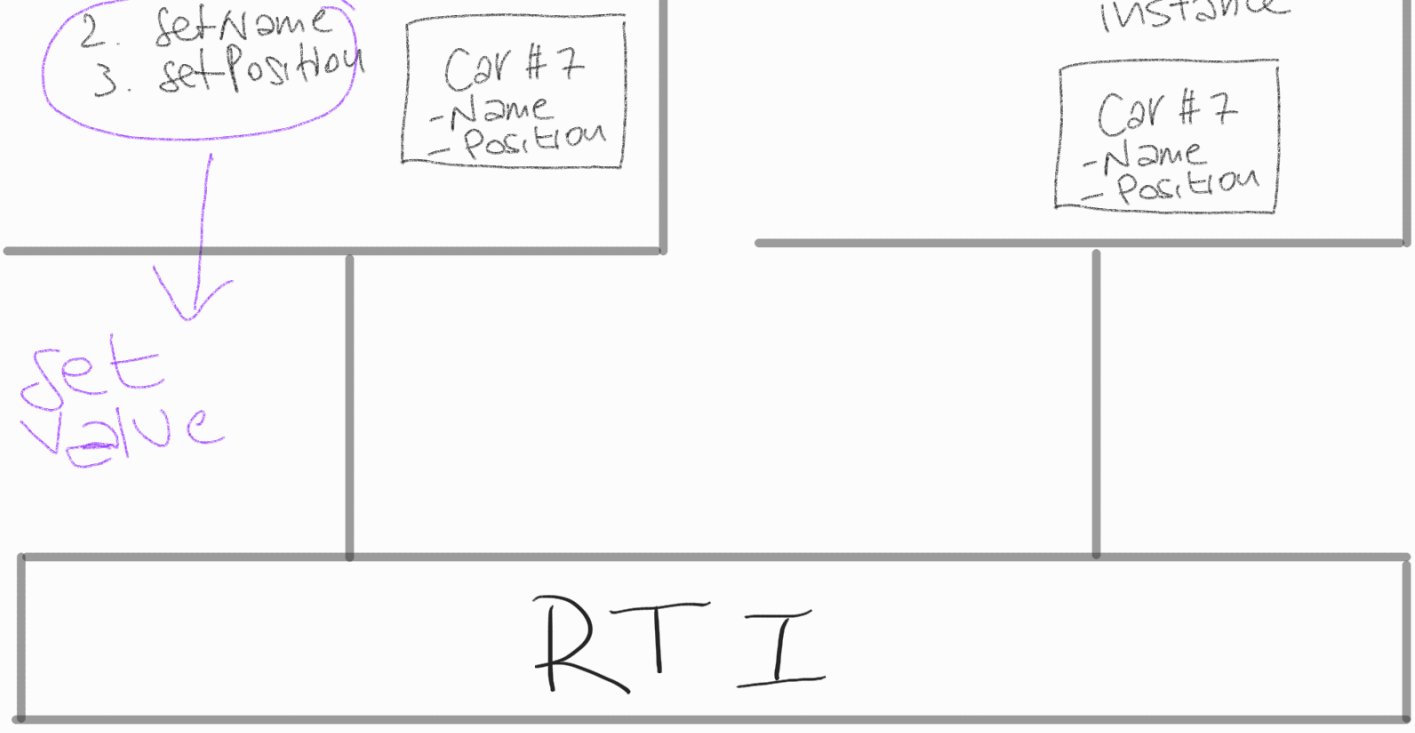
Is not a standard. It's design pattern for the HLA module of an application. The object instances were registered by the federate that I want.. and I can get the values of the instance through a "get" operation. The federate can creates his own local object instances and the module then will register the object in the federation. It's possible updates the value of attributes with the "set balue" operation.
ADV:

- Fast and easy to add an HLA module
- No to learn all HLa system
- Reduce risk of incorrect data encoding and decoding

DIS:

- The middleware is locked to one FOM. It's not suitable for general purposes tools that need to handle different FOMs
- Need to acquire Object Oriented HLA module
- Handle Configuration in the way it cannot subscribe mode object class than necessary





MESSAGES HLA

Updates : send and receive new attributes values for each time step ending

Interaction: send and receive parameters describing events when happens. Interaction are temporary.

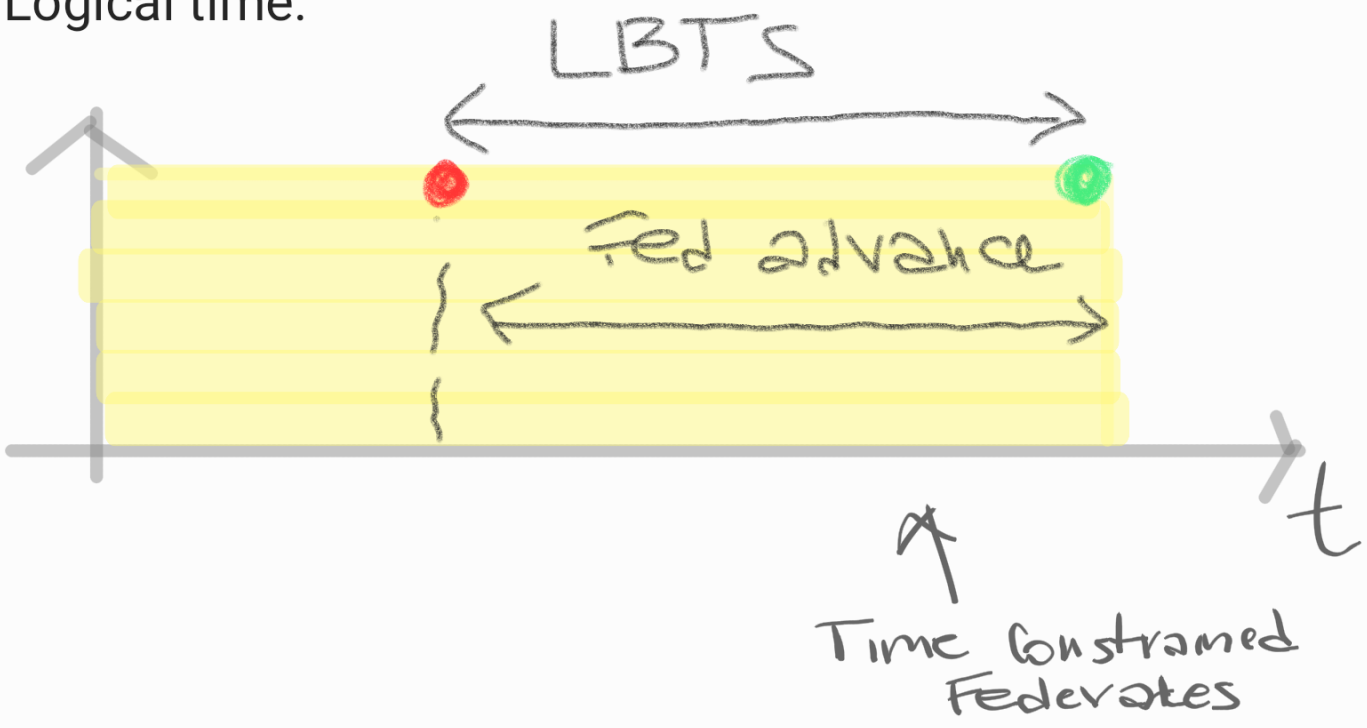
Subscription and pubblication: I can subscribe to an object that is publication or I can publish an object. Registration is for instance of objects. For Interaction everything is temporary as Interaction itself.

Two mesage order type:

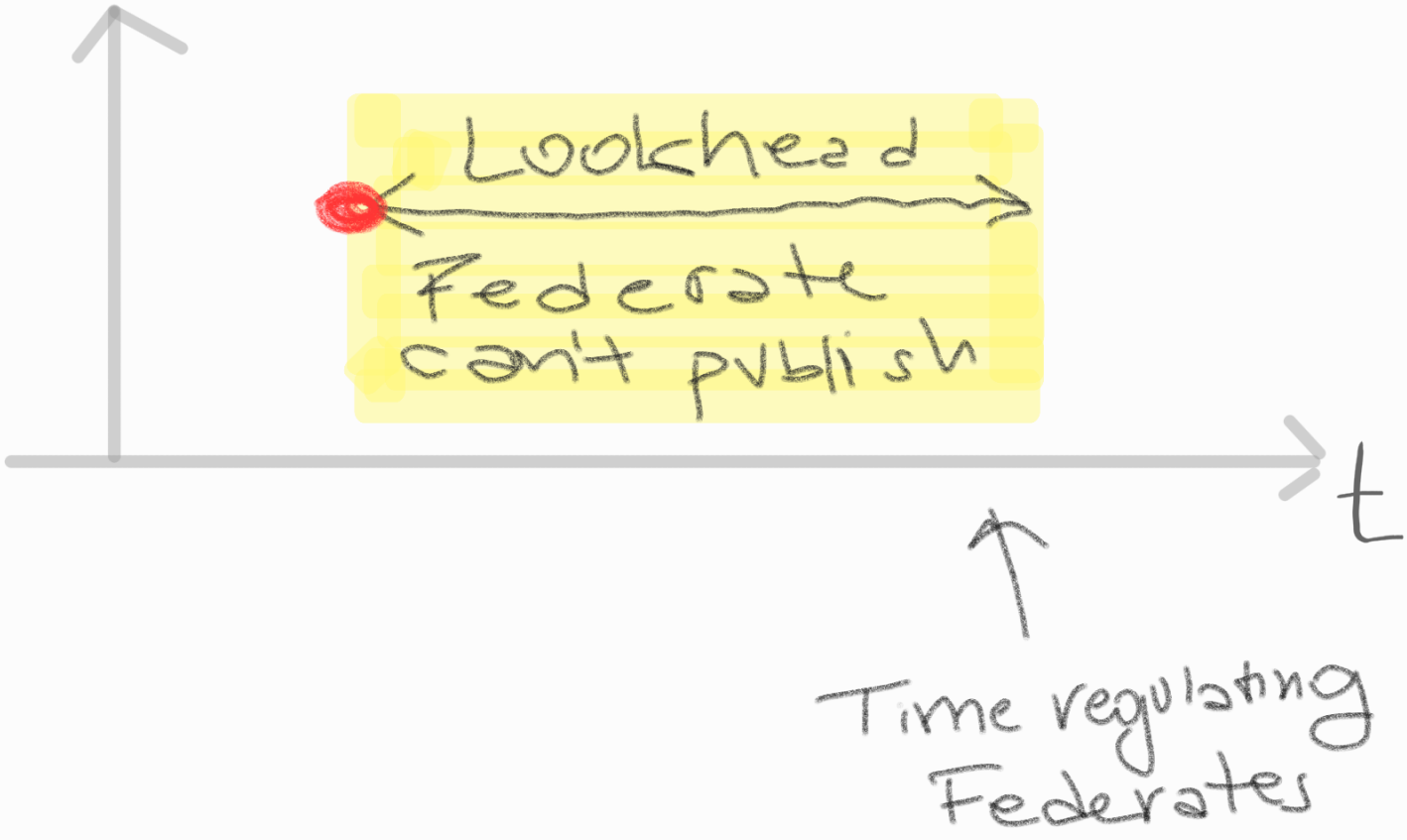
- TSO: time stamp order, message delivered to federate in the indicated order
- RO: receive order, message delivered to federate in the sent order

Time constraints and time regulating influenza l' ordine di ricezione dei messaggi. La federate deve dichiarare questi due parametri per ricevere o inviare TSO.

Logical time:



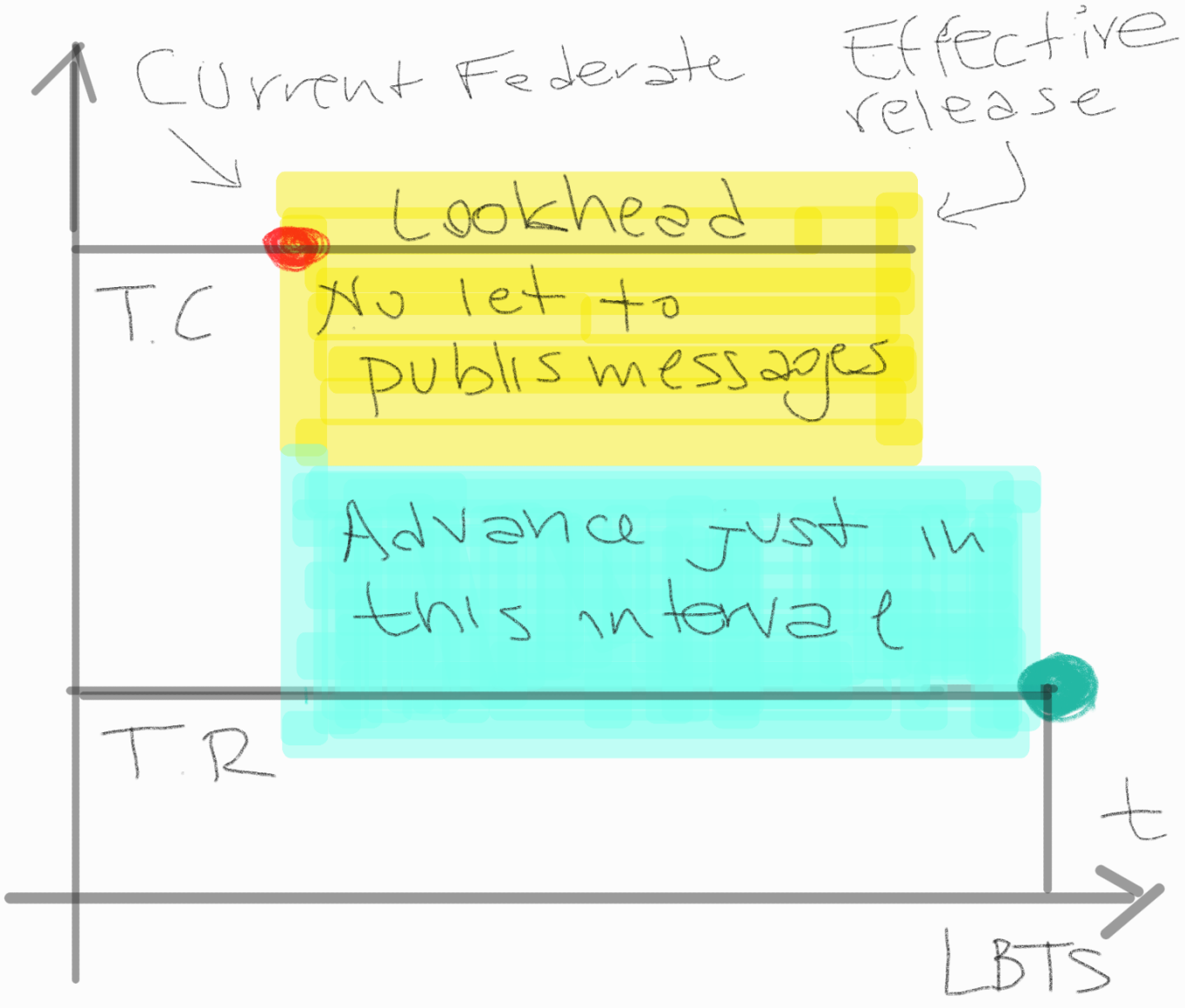
LBTS: lower bound time stamp



TIME ADVANCE:
In continuos simulation we use Time advance request using RTI service "request time advance" --> time advance Grant is notified

In discrete event simulation --> we have "next event request" and is notified the "next event grant " to go forward.

Federates send and receive TSO data, must be TR and TC.



TIME MANAGEMENT
types of Simulator : continuos and discrete

Types of simulation implementations:

- Event driven implementation
- Time stepped implementation
- Real Time implementation
- Scaled real time implementation
- Non real time implementation

Time stepped example: continous model of a discrete process, uses events to communicate existence of countries. Original model built, it's assumed that each federate start at time 0.

Non existence of scheduling del time management. Each federate advances time at its own pace.
Conservative:

- syncronization: Federates advance time only when guaranteed that no past data will be received
- Optimistic synchronization : free to advance logical time, may have a Rollback.
- Active scan : advance time by mutual agreement with other federates.

Logical time restriction:

- Initial value
- Value not tied to any Measure units system
- Well ordered
- Always greater or equal than initial time
- The time is effectively discrete
- A special value exists, it' s called Positive Inifinte and it's greate than any other value

Logical time synchronization:

Case1
Update delayed until clock advances to end of time stamp.

Case2
Update delayed until clock advances to end of time stamp.

Events:
Sending events: Update attributes value, send interaction, delete object instance.
Receiving events: reflect attribute values, interaction and object instance.