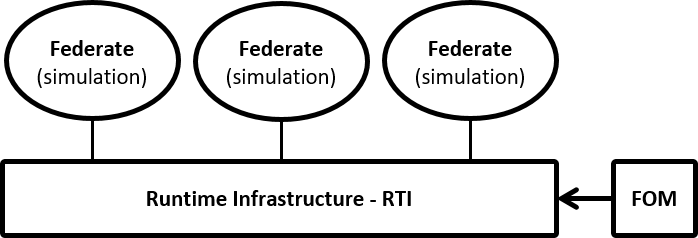
# User guide of a distributed system based on the comunication protocol HLA

## HLA

HLA (High Level Architecture) is an open international standard, developed by the Simulation Interoperability Standards Organization (SISO) and published by IEEE.

Essentially, HLA is a comunication protocol used to develop distributed simulation. The simulations are called federates; each federate is able to exchange data to each other thanks to the interoperability of HLA.

Here the Architecture:



The component of HLA are:

1. **FEDERATES**: simulations, Including a wide types of simulation software and/or normal Java programs developed to be a simulation.

2. **RTI**:(Runtime Infrastructure). It's the linker between federates. It’s regulated by FOM file and thanks to it the simulations can receive and send data among them. For example, if a federate wants to send a signal as interaction to start the simulation in another federate, can send the interaction through RTI and the other federate receive the interaction. The second federate send a Callback once has receveid the data to notify it, always passing through the RTI.

3. **FOM** (Federation object model) is a ".xml" file, where is descripted the structure of data trasmitted through the RTI. The data can be:

*\*Interactions;*

*\*Objects.*

In the FOM the data are specified in term of Interaction handle and Parameter handle for the interactions data, while for object data the data type are Object handle and attribute handle.

Moreover, the file contains also other information about the federation as well as the federation name, federates names and others.

4. **AMBASSADOR**: internal component of the architecture. The ambassador is double, one is located in the federates and one in the RTI. The two ambassadors make the CALL and the CALLBACKS between federates. It's the core of the comunication protocol.

5. **FEDERATION**: the federation is the set of components over mentioned.

## Software Used

### Java:

I've implemented 3 modules in Java code:

- coordinator

- hlacore

- master

Each modules has a specific porpouse and a different way of implementation. Hlacore and coordinator are implemented in pure Java.

Master is developed with Spring Boot framework, which allows to build a REST service for a web app. That's why the need of POSTMAN, that I'll discuss in details later.

***hlaCore*** is a jar library containing the functions that "knows" HLA, that is to say the functions that calls method of hla library on the ambassador. The building of this library is the key of scalability, this module is, indeed, reusable in other projects in a more easy and less hard working way.

***Coordinator*** is a jar library that coordinate the calls and callback from a simulation model and the hlaCore. It's easily re-adaptable to every simulation models. In this case, I used AnyLogic to simulate the route of a car on the scenario starts from Cupertino and arrived to Pacifica.

The car, as written in the FOM file, can send and receive interactions, like "load scenario" and "start & stop" simulation. Moreover, it can receive and sends data about the car, like what "type of fuel" the car is refueled or the "fuel level" of it.

This library has to be imported in the software simulation to be used.

***Manager*** is a federate (module) that control the other federates. As just said, it's a little web service that works with RESTFUL API calls. No needs to export it as modules in a java library because it's not imported anywhere. The controller handles the HLA functionality, like the connection between federate and RTI in the FedExec.

### POSTAMN

Postman is an API platform used for designing, testing and developing APIs.

In this project, only Master federate is a web service that need to use POSTMAN.

I have built a workspace in which put the API's calls.

The GET calls are:

\* GET init to connect and join the federate to pitch which was activated in advance.

\* GET exit to disconnet and shut the federate linked to pitch.

\* GET startInteraction that takes also a parameter (TimeScaleFactor) to get the start interaction from pitch.

\* GET stopInteraction to get the stop interaction form pitch.

And the POST call is:

\* POST injectCar that post a Car with 3 attributes (name, license plate and color) into AnyLogic model.

### AnyLogic

AnyLogic is a tool to create simulation models, developed by The AnyLogic Company. AnyLogic is implemented in Java and use Java language to behave the blocks of the model and the model itself; the blocks are virtual components listed in the AnyLogic's palette (which seems a library of components).

How to link Anylogic to the Master federate? Simply importing the coordinator.jar library to the project model in use. Now, the model can employ and override the methods of the library.

### Pitch

PITCH is the HLA environments in which the federation is created. It mainly let all the components of a federation to join and create the FedExec.

Actually, PITCH is a GUI to visualize the federation and the interaction exchanged by every federates.

## **MANUALS**

## Installation

To install PITCH:

\* It's available a guide written by authors of PITCH.

Other module to export:

\* As just said the coordinator.jar library is the one who behave the model, so it's should be imported in AnyLogic model in the section dedicated;

\* The others modules to import in AnyLogic model are: hlaCore.jar, which is now in a local maven repository; and the library of open-hla, prti1516e.jar.

## Run

The steps to run the program until now are:

1. Open Pitch

2. Run Master federate: this one has the task to create the federation in Pitch environment and then join itself.

3. Run AnyLogic simulation with Play button. At this point the federation is completed; all the members has joined.

4. Open POSTMAN: and run the calls of interest. In this case: GET loadScenario to load the Scenario into the federates and then GET startInteraction to start the interaction. In the end launch the simulation with POST injectCar.