International Summer School on Industrial Agents 2024

Standardization of I4.0 Systems

Multi-agent programming with SPADE
Part II: Gateway Agent
Part III: IEEE 2660.1

Oskar Casquero, Aintzane Armentia, Julen Cuadra, Ekaitz Hurtado

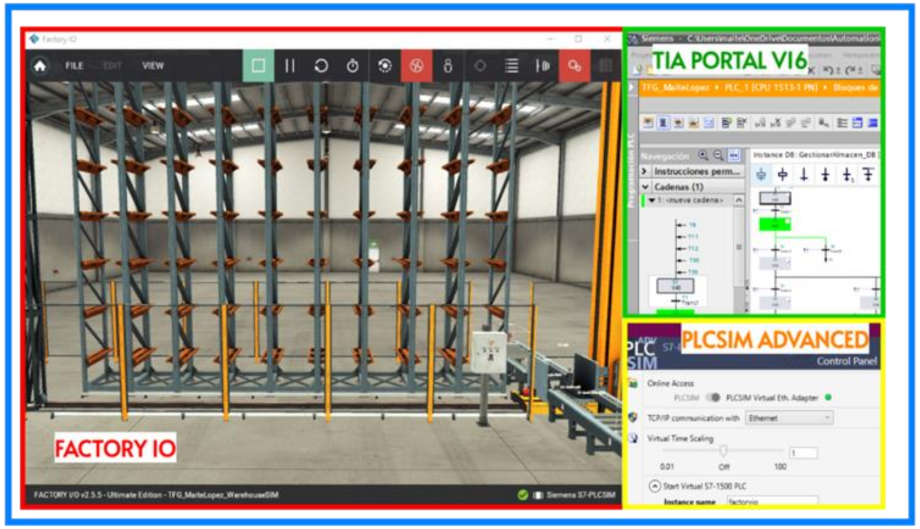
Bilbao, 24th June 2024

SMALL USE CASE

INTRODUCTION

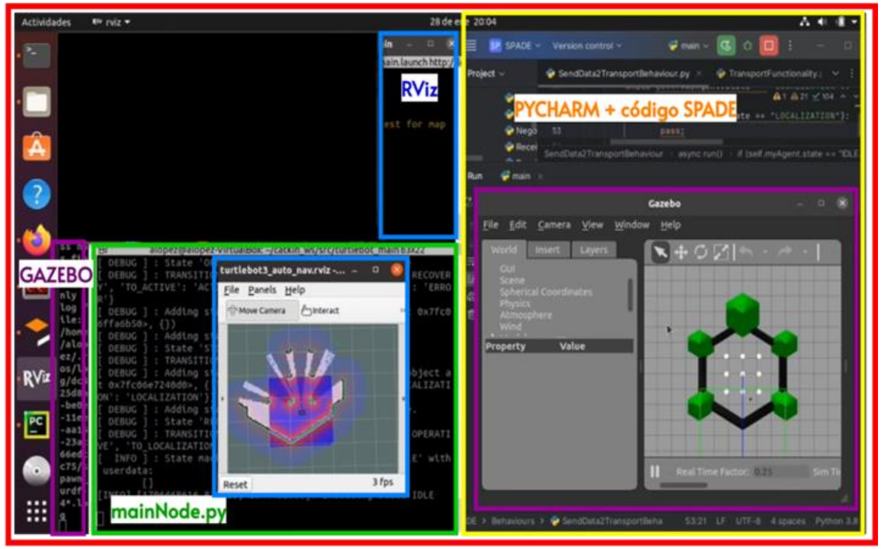
- Two type of physical assets: a warehouse and a transport robot
- Need to coordinate smart warehouse with transportation robot

WAREHOUSE



Oskar's computer

TRANSPORT ROBOT



Your computer

SMALL USE CASE

INTRODUCTION

- Two type of physical assets: a warehouse and a transport robot
  - Service 1: DELIVERY
    - 1 machine agent (warehouse) and 1 transport agent (turtlebot)
    - The process is initiated by the transport agent.
    - When the robot arrives to the warehouse, the transport agent notifies the machine agent.
  - Service 2: COLLECTION
    - 1 machine agent (warehouse) and 2 transport agent (turtlebot)
    - The process is initiated by the machine agent.
    - When the parcel is at the exit point of the warehouse, the machine agent request a transport service.
    - The transport agents perform a distributed negotiation (one with each other, without the intervention of a central entity).
    - The winner performs the transportation service.

SMALL CASE STUDY

INTRODUCTION

- Start the environment in your computer:
  - Open a **Terminator** window.
  - In the Terminator window, change directory to **turtlebot_main** ROS package developed for this case study: **roscd turtlebot_main**
  - Split the Terminator window horizontally to create another two terminals:
  - In each of the terminals:
    - Terminal 1. Start Gazebo: **./scripts/initGazebo.sh**

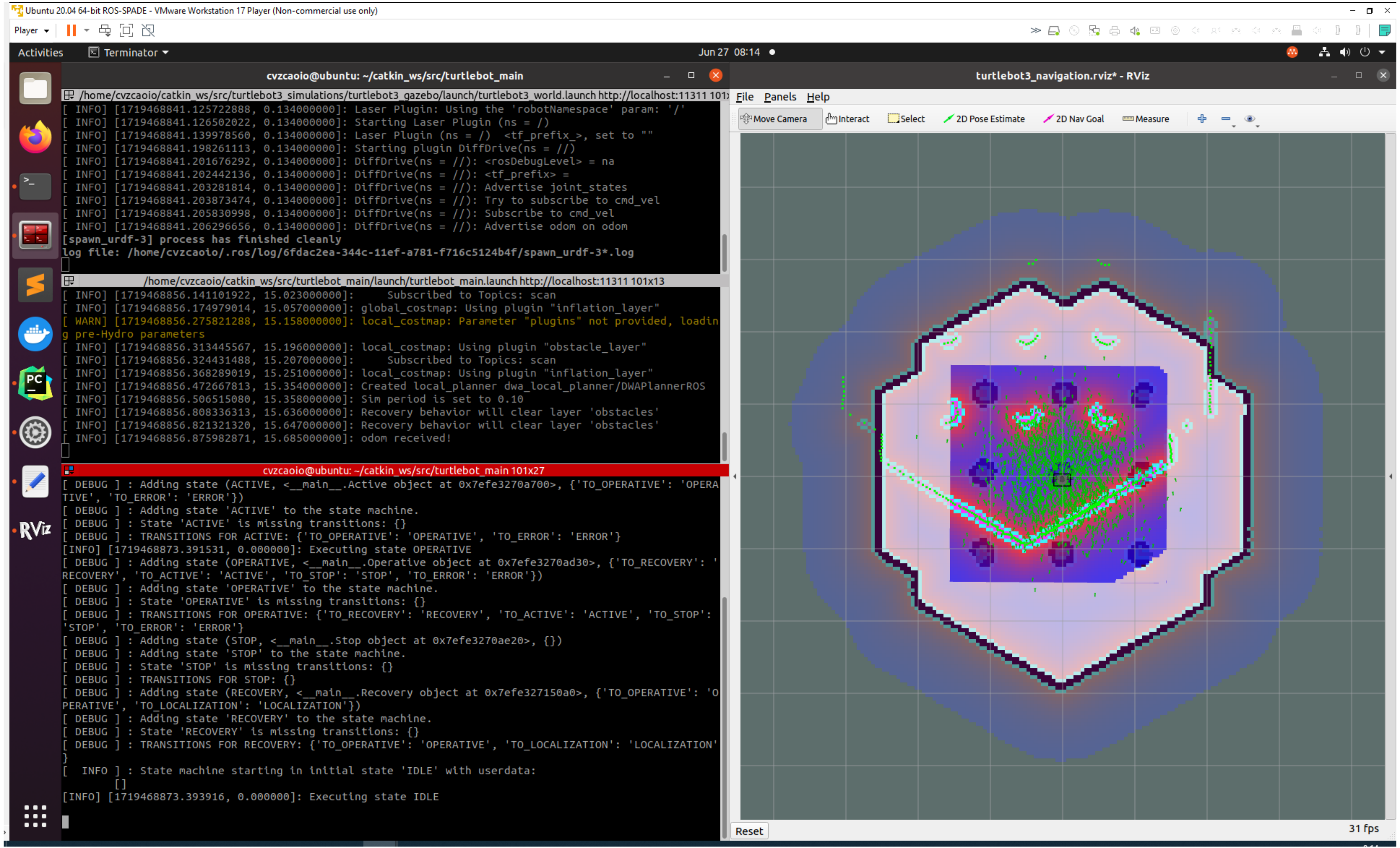
      Gazebo is a 3D robot simulator. Its objective is to simulate a robot, giving you a close substitute to how your robot would behave in a real-world physical environment.
    - Terminal 2. Start Rviz: **./scripts/initRviz.sh**

      rviz (short for "ROS visualization") is a 3D visualization tool for robots, sensors, and algorithms. It enables to see the robot's perception of its world (real or simulated).

      Do 2D pose estimation to position and direct the robot in the simulated environment.
    - Terminal 3. Start the transport robot main node (developed specifically for this project): **rosrun turtlebot_main mainNode.py**

SMALL CASE STUDY

INTRODUCTION


Ubuntu 20.04 64-bit ROS-SPADE - VMware Workstation 17 Player (Non-commercial use only)
Player
Activities
Terminator
Jun 27 08:14
cvzcaoio@ubuntu: ~/catkin_ws/src/turtlebot_main
turtlebot3_navigation.rviz* - RViz
File
Panels
Help
Move Camera
Interact
Select
2D Pose Estimate
2D Nav Goal
Measure
Reset
31 fps
RViz

/home/cvzcaoio/catkin_ws/src/turtlebot3_simulations/turtlebot3_gazebo/launch/turtlebot3_world.launch http://localhost:11311 101x

```
[ INFO] [1719468841.125722888, 0.134000000]: Laser Plugin: Using the 'robotNamespace' param: '/'
[ INFO] [1719468841.126502022, 0.134000000]: Starting Laser Plugin (ns = /)
[ INFO] [1719468841.139978560, 0.134000000]: Laser Plugin (ns = /)  <tf_prefix_>, set to ""
[ INFO] [1719468841.198261113, 0.134000000]: Starting plugin DiffDrive(ns = //)
[ INFO] [1719468841.201676292, 0.134000000]: DiffDrive(ns = //): <rosDebugLevel> = na
[ INFO] [1719468841.202442136, 0.134000000]: DiffDrive(ns = //): <tf_prefix> =
[ INFO] [1719468841.203281814, 0.134000000]: DiffDrive(ns = //): Advertise joint_states
[ INFO] [1719468841.203873474, 0.134000000]: DiffDrive(ns = //): Try to subscribe to cmd_vel
[ INFO] [1719468841.205830998, 0.134000000]: DiffDrive(ns = //): Subscribe to cmd_vel
[ INFO] [1719468841.206296656, 0.134000000]: DiffDrive(ns = //): Advertise odom on odom
[spawn_urdf-3] process has finished cleanly
log file: /home/cvzcaoio/.ros/log/6fdac2ea-344c-11ef-a781-f716c5124b4f/spawn_urdf-3*.log
```

/home/cvzcaoio/catkin_ws/src/turtlebot_main/launch/turtlebot_main.launch http://localhost:11311 101x13

```
[ INFO] [1719468856.141101922, 15.023000000]:     Subscribed to Topics: scan
[ INFO] [1719468856.174979014, 15.057000000]: global_costmap: Using plugin "inflation_layer"
[ WARN] [1719468856.275821288, 15.158000000]: local_costmap: Parameter "plugins" not provided, loadin
g pre-Hydro parameters
[ INFO] [1719468856.313445567, 15.196000000]: local_costmap: Using plugin "obstacle_layer"
[ INFO] [1719468856.324431488, 15.207000000]:     Subscribed to Topics: scan
[ INFO] [1719468856.368289019, 15.251000000]: local_costmap: Using plugin "inflation_layer"
[ INFO] [1719468856.472667813, 15.354000000]: Created local_planner dwa_local_planner/DWAPlannerROS
[ INFO] [1719468856.506515080, 15.358000000]: Sim period is set to 0.10
[ INFO] [1719468856.808336313, 15.636000000]: Recovery behavior will clear layer 'obstacles'
[ INFO] [1719468856.821321320, 15.647000000]: Recovery behavior will clear layer 'obstacles'
[ INFO] [1719468856.875982871, 15.685000000]: odom received!
```

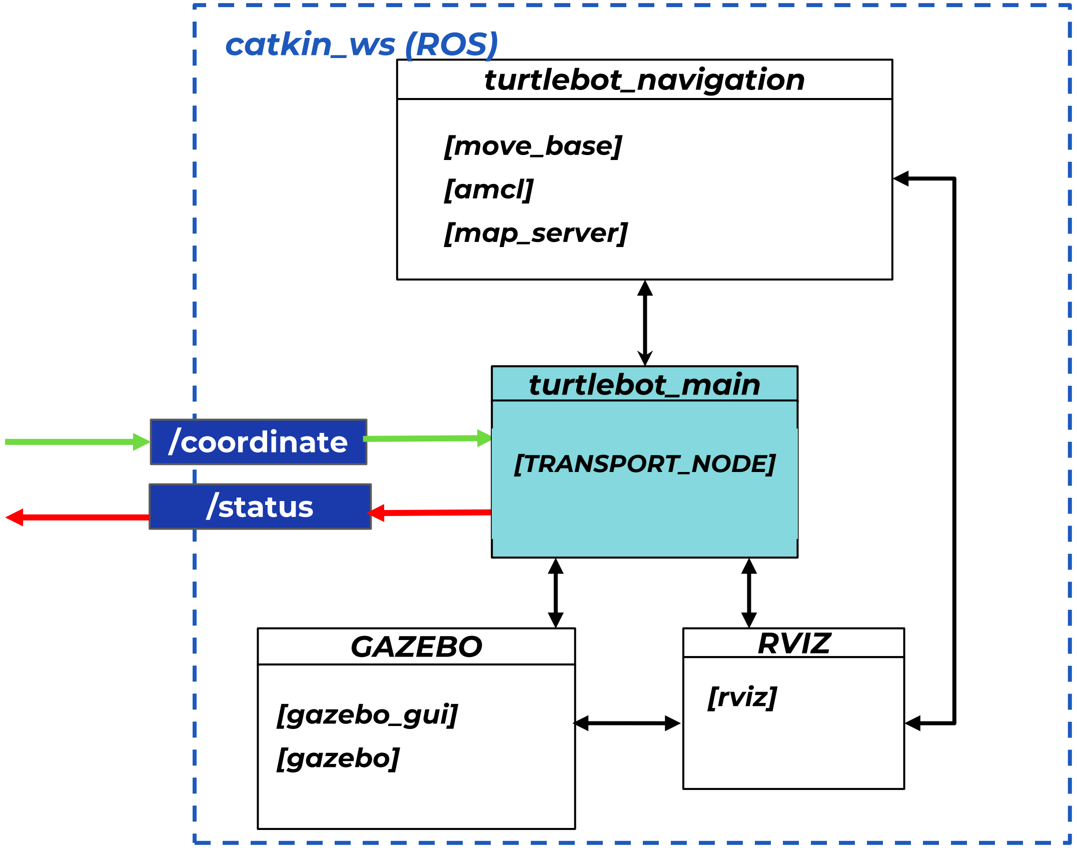
cvzcaoio@ubuntu: ~/catkin_ws/src/turtlebot_main 101x27

```
[ DEBUG ] : Adding state (ACTIVE, <__main__.Active object at 0x7efe3270a700>, {'TO_OPERATIVE': 'OPERA
TIVE', 'TO_ERROR': 'ERROR'})
[ DEBUG ] : Adding state 'ACTIVE' to the state machine.
[ DEBUG ] : State 'ACTIVE' is missing transitions: {}
[ DEBUG ] : TRANSITIONS FOR ACTIVE: {'TO_OPERATIVE': 'OPERATIVE', 'TO_ERROR': 'ERROR'}
[INFO] [1719468873.391531, 0.000000]: Executing state OPERATIVE
[ DEBUG ] : Adding state (OPERATIVE, <__main__.Operative object at 0x7efe3270ad30>, {'TO_RECOVERY': '
RECOVERY', 'TO_ACTIVE': 'ACTIVE', 'TO_STOP': 'STOP', 'TO_ERROR': 'ERROR'})
[ DEBUG ] : Adding state 'OPERATIVE' to the state machine.
[ DEBUG ] : State 'OPERATIVE' is missing transitions: {}
[ DEBUG ] : TRANSITIONS FOR OPERATIVE: {'TO_RECOVERY': 'RECOVERY', 'TO_ACTIVE': 'ACTIVE', 'TO_STOP':
'STOP', 'TO_ERROR': 'ERROR'}
[ DEBUG ] : Adding state (STOP, <__main__.Stop object at 0x7efe3270ae20>, {})
[ DEBUG ] : Adding state 'STOP' to the state machine.
[ DEBUG ] : State 'STOP' is missing transitions: {}
[ DEBUG ] : TRANSITIONS FOR STOP: {}
[ DEBUG ] : Adding state (RECOVERY, <__main__.Recovery object at 0x7efe327150a0>, {'TO_OPERATIVE': 'O
PERATIVE', 'TO_LOCALIZATION': 'LOCALIZATION'})
[ DEBUG ] : Adding state 'RECOVERY' to the state machine.
[ DEBUG ] : State 'RECOVERY' is missing transitions: {}
[ DEBUG ] : TRANSITIONS FOR RECOVERY: {'TO_OPERATIVE': 'OPERATIVE', 'TO_LOCALIZATION': 'LOCALIZATION'
}
[  INFO ] : State machine starting in initial state 'IDLE' with userdata:
        []
[INFO] [1719468873.393916, 0.000000]: Executing state IDLE
```

SMALL CASE STUDY

INTRODUCTION

- Transport robot main node (developed specifically for this project)



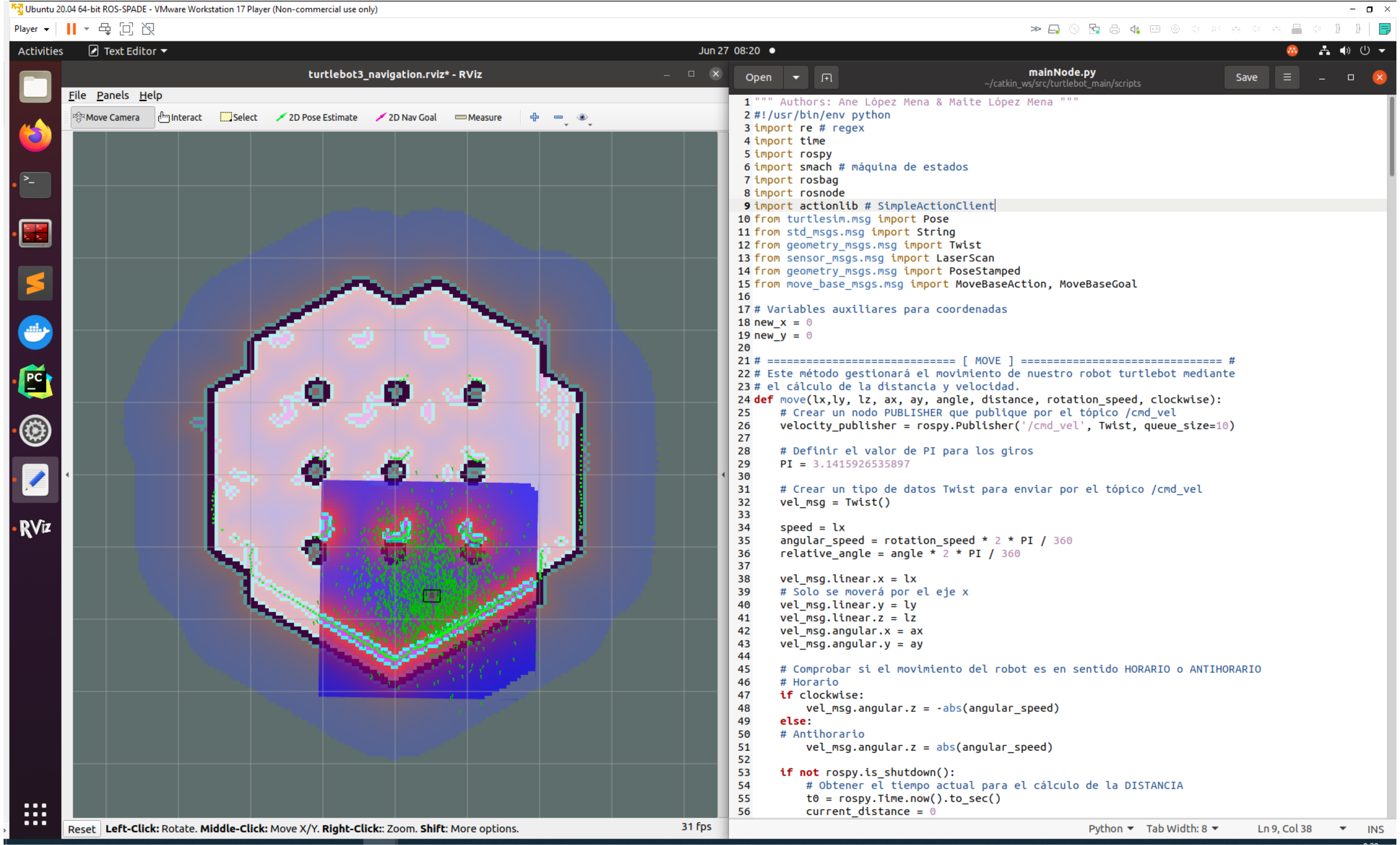
It implements:

- An interface based on two ROS topics to interact with the robot:
  - sending moving coordinates
  - reading its status
- A Finite State Machine to control the robot and coordinate it with the nodes of the navigation stack, gazebo and Rviz.

You can read its code using the **text editor** from the **applications menu of Ubuntu** and opening “mainNode.py” from recent files list.

SMALL CASE STUDY

INTRODUCTION


Ubuntu 20.04 64-bit ROS-SPADE - VMware Workstation 17 Player (Non-commercial use only)
Player
Activities
Text Editor
Jun 27 08:20
turtlebot3_navigation.rviz* - RViz
File
Panels
Help
Move Camera
Interact
Select
2D Pose Estimate
2D Nav Goal
Measure
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Zoom. Shift: More options.
31 fps
Open
Save
mainNode.py
~/catkin_ws/src/turtlebot_main/scripts
Python
Tab Width: 8
Ln 9, Col 38
INS

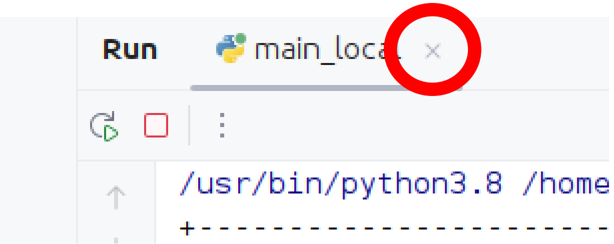
```python
""" Authors: Ane López Mena & Maite López Mena """
#!/usr/bin/env python
import re # regex
import time
import rospy
import smach # máquina de estados
import rosbag
import rosnode
import actionlib # SimpleActionClient
from turtlesim.msg import Pose
from std_msgs.msg import String
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan
from geometry_msgs.msg import PoseStamped
from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal

# Variables auxiliares para coordenadas
new_x = 0
new_y = 0

# ============================= [ MOVE ] =============================== #
# Este método gestionará el movimiento de nuestro robot turtlebot mediante
# el cálculo de la distancia y velocidad.
def move(lx,ly, lz, ax, ay, angle, distance, rotation_speed, clockwise):
    # Crear un nodo PUBLISHER que publique por el tópico /cmd_vel
    velocity_publisher = rospy.Publisher('/cmd_vel', Twist, queue_size=10)

    # Definir el valor de PI para los giros
    PI = 3.1415926535897

    # Crear un tipo de datos Twist para enviar por el tópico /cmd_vel
    vel_msg = Twist()

    speed = lx
    angular_speed = rotation_speed * 2 * PI / 360
    relative_angle = angle * 2 * PI / 360

    vel_msg.linear.x = lx
    # Solo se moverá por el eje x
    vel_msg.linear.y = ly
    vel_msg.linear.z = lz
    vel_msg.angular.x = ax
    vel_msg.angular.y = ay

    # Comprobar si el movimiento del robot es en sentido HORARIO o ANTIHORARIO
    # Horario
    if clockwise:
        vel_msg.angular.z = -abs(angular_speed)
    else:
    # Antihorario
        vel_msg.angular.z = abs(angular_speed)

    if not rospy.is_shutdown():
        # Obtener el tiempo actual para el cálculo de la DISTANCIA
        t0 = rospy.Time.now().to_sec()
        current_distance = 0
```

SMALL CASE STUDY

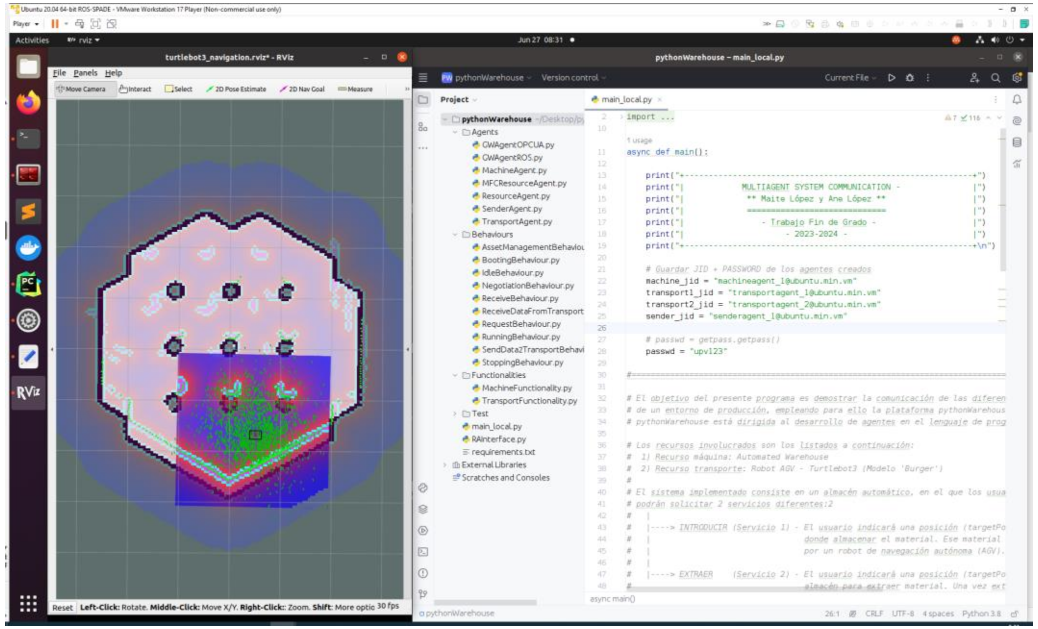
INTRODUCTION

- Start the environment in your computer:
  - Open a **Terminal** window (not a Terminator window) and execute the following command to open PyCharm: **./pycharm-2023.3/bin/pycharm.sh**
  - A project called SPADE should be automatically opened. If not, open it from recent projects list (PyCharm → File → Recent Projects).
  - Expand the project tree to see all the contents of the project.
  - Open **main_local.py**
  - Execute main_local.py

    NOTE: to rerun main_local.py you should click here:



```
/usr/bin/python3.8 /home
+-----------------------
```



```
import ...

1 usage
async def main():

    print("+---------------------------------------------------------------+")
    print("|              MULTIAGENT SYSTEM COMMUNICATION -                |")
    print("|               ** Maite López y Ane López **                   |")
    print("|               ==============================                  |")
    print("|                   - Trabajo Fin de Grado -                    |")
    print("|                        - 2023-2024 -                          |")
    print("+---------------------------------------------------------------+\n")

    # Guardar JID + PASSWORD de los agentes creados
    machine_jid = "machineagent_1@ubuntu.min.vm"
    transport1_jid = "transportagent_1@ubuntu.min.vm"
    transport2_jid = "transportagent_2@ubuntu.min.vm"
    sender_jid = "senderagent_1@ubuntu.min.vm"

    # passwd = getpass.getpass()
    passwd = "upv123"

#==========================================================================

# El objetivo del presente programa es demostrar la comunicación de las diferen
# de un entorno de producción, empleando para ello la plataforma pythonWarehous
# pythonWarehouse está dirigida al desarrollo de agentes en el lenguaje de prog

# Los recursos involucrados son los listados a continuación:
#  1) Recurso máquina: Automated Warehouse
#  2) Recurso transporte: Robot AGV - Turtlebot3 (Modelo 'Burger')
#
# El sistema implementado consiste en un almacén automático, en el que los usua
# podrán solicitar 2 servicios diferentes:2
#   |
#   |----> INTRODUCIR (Servicio 1) - El usuario indicará una posición (targetPo
#   |                                donde almacenar el material. Ese material
#   |                                por un robot de navegación autónoma (AGV).
#   |
#   |----> EXTRAER    (Servicio 2) - El usuario indicará una posición (targetPo
#                                    almacén para extraer material. Una vez ext
async main()
```

ISSUE

INTRODUCTION

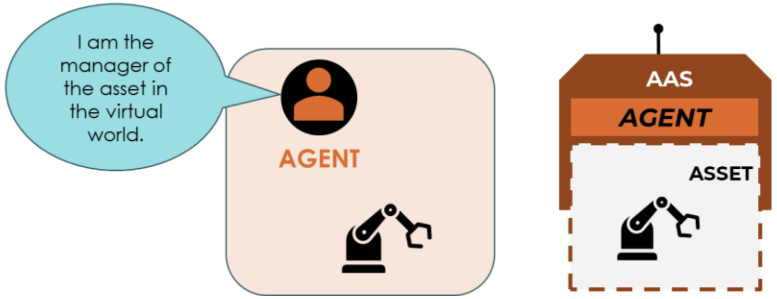
- Two type of physical assets: a warehouse and a transport robot
- Need to orchestrate their operation for each service.

- How can we do this SPADE? Proposals? Resources:
  - A SPADE agent is an instance of a Python class
    - The class is inherited from a SPADE class: **spade.agent.Agent**
    - **setup()**: to add initialization code
  - Behaviour: pattern-based task executed by an agent
    - A Behaviour is an instance of a Python class inherited from a SPADE class:
      - **spade.behaviour.CyclicBehaviour**
      - **spade.behaviour.OneShotBehaviour**
      - **spade.behaviour.FSMBehaviour**
  - Behaviour common methods
    - **run()**: where the core of the logic is executed
  - SPADE agents can communicate with each other by exchanging **messages**.
  - **Templates** are used to automatically dispatch received messages to the behaviours that are waiting for them.

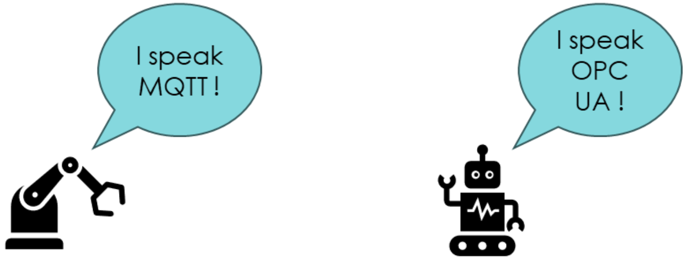
ASSET INTEGRATION

INTRODUCTION

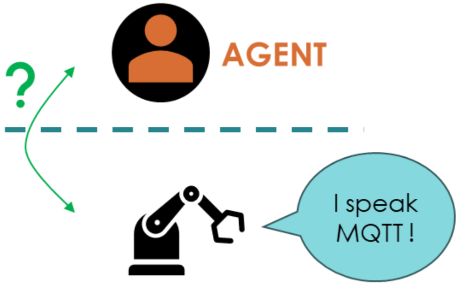
- Agents can:
  - add intelligence to assets
  - manage service requests related to those assets
  - implement Asset Administration Shells



- But each asset has different communication capabilities

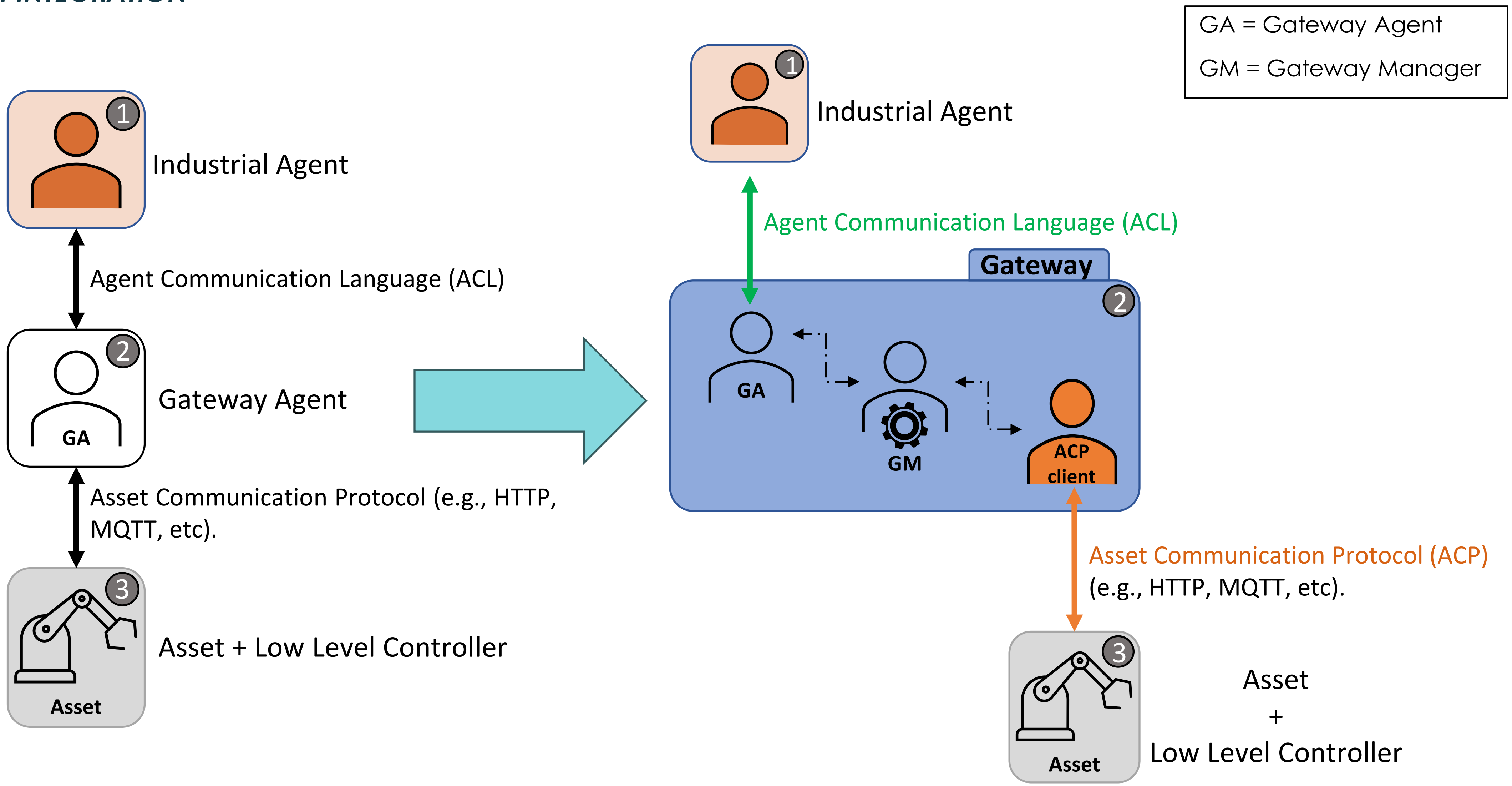


- How can we access physical assets in a common way?



GATEWAY DESIGN PATTERN

PHYSICAL ASSET INTEGRATION



GA = Gateway Agent

GM = Gateway Manager

1 Industrial Agent

Agent Communication Language (ACL)

2 GA Gateway Agent

Asset Communication Protocol (e.g., HTTP, MQTT, etc).

3 Asset Asset + Low Level Controller

1 Industrial Agent

Agent Communication Language (ACL)

Gateway 2

GA

GM

ACP client

Asset Communication Protocol (ACP) (e.g., HTTP, MQTT, etc).

3 Asset

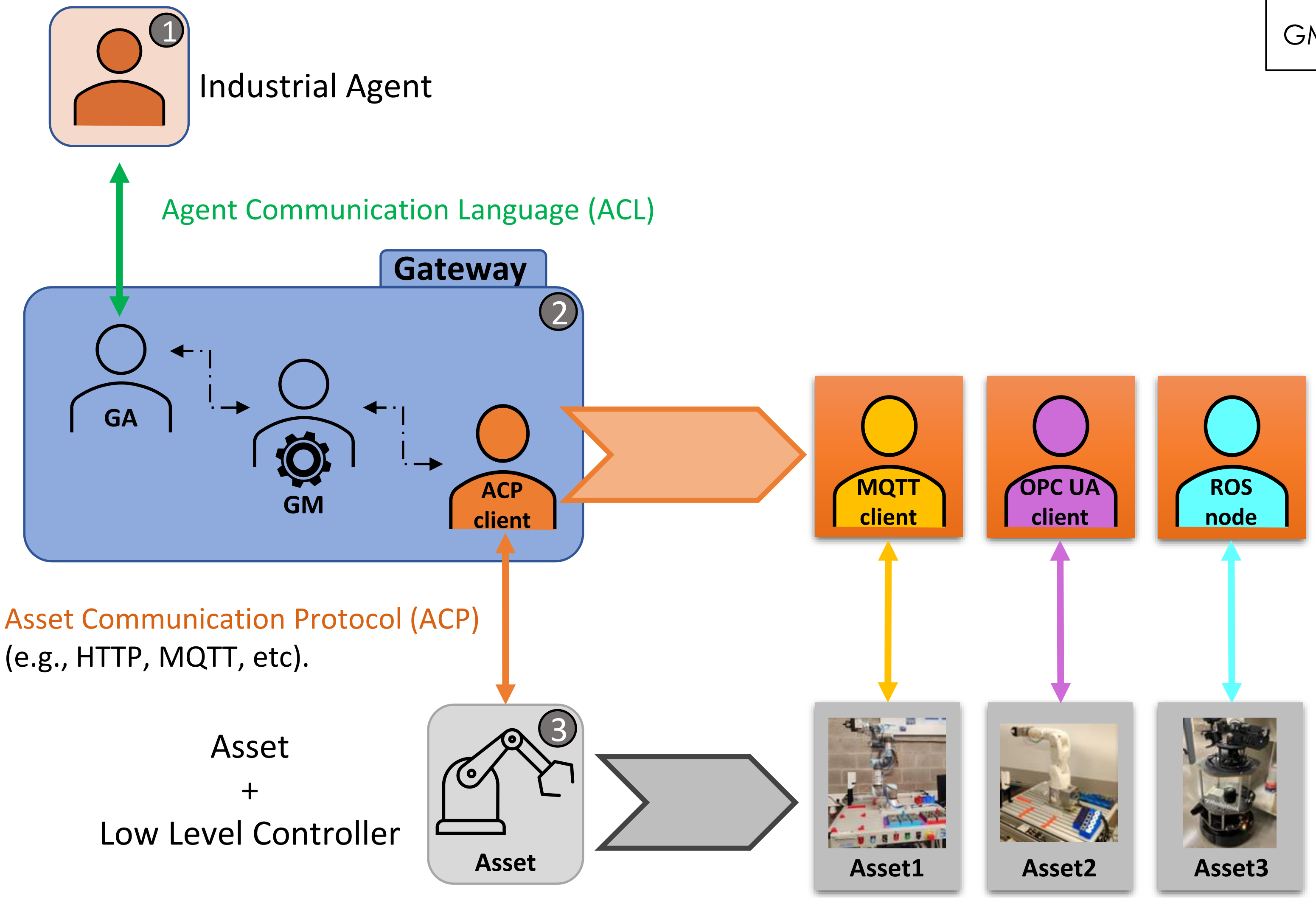
Asset
+
Low Level Controller

GATEWAY DESIGN PATTERN

PHYSICAL ASSET INTEGRATION

GA = Gateway Agent

GM = Gateway Manager



Industrial Agent

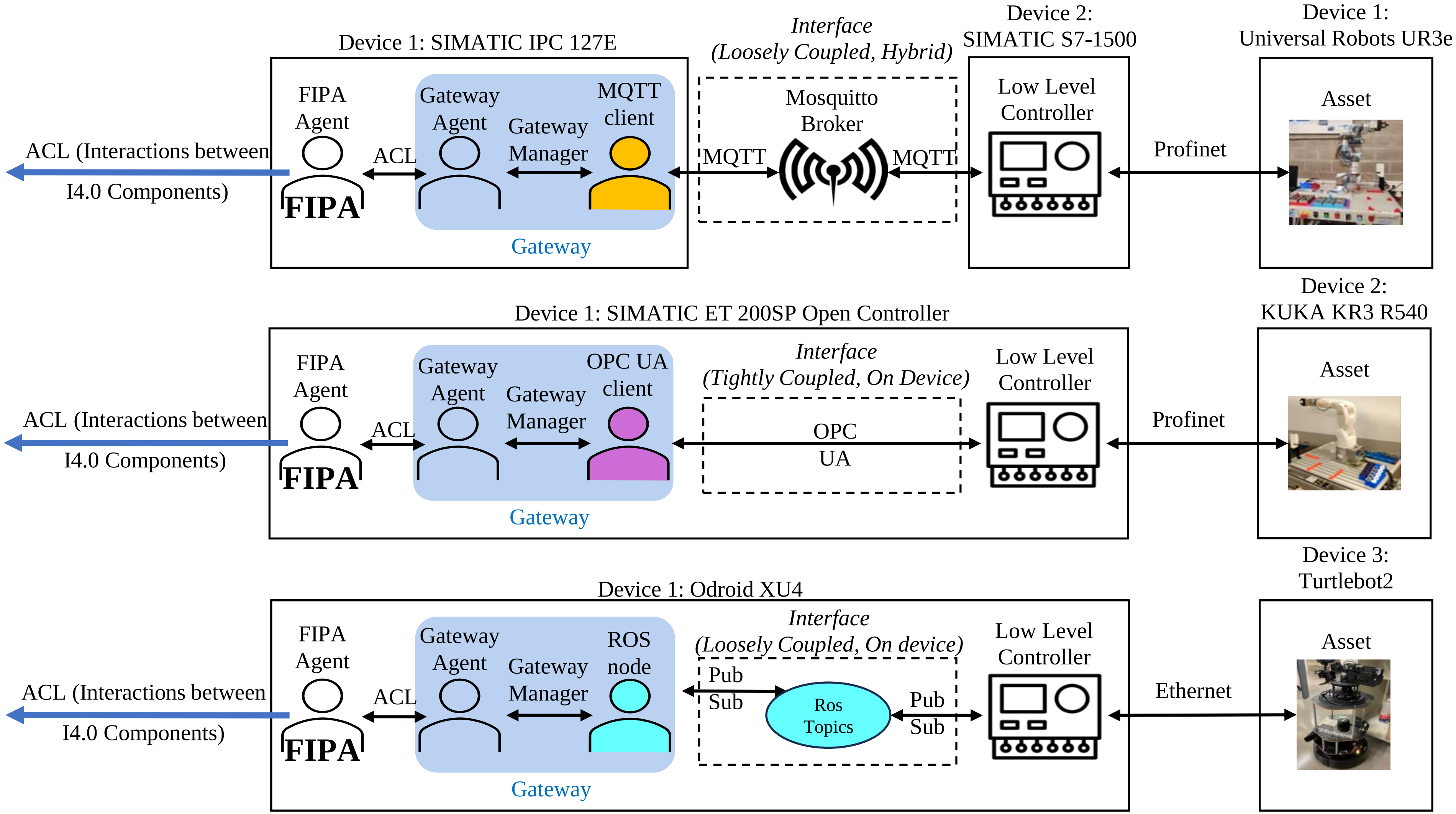
Agent Communication Language (ACL)

Asset Communication Protocol (ACP)
(e.g., HTTP, MQTT, etc).

Asset
+
Low Level Controller

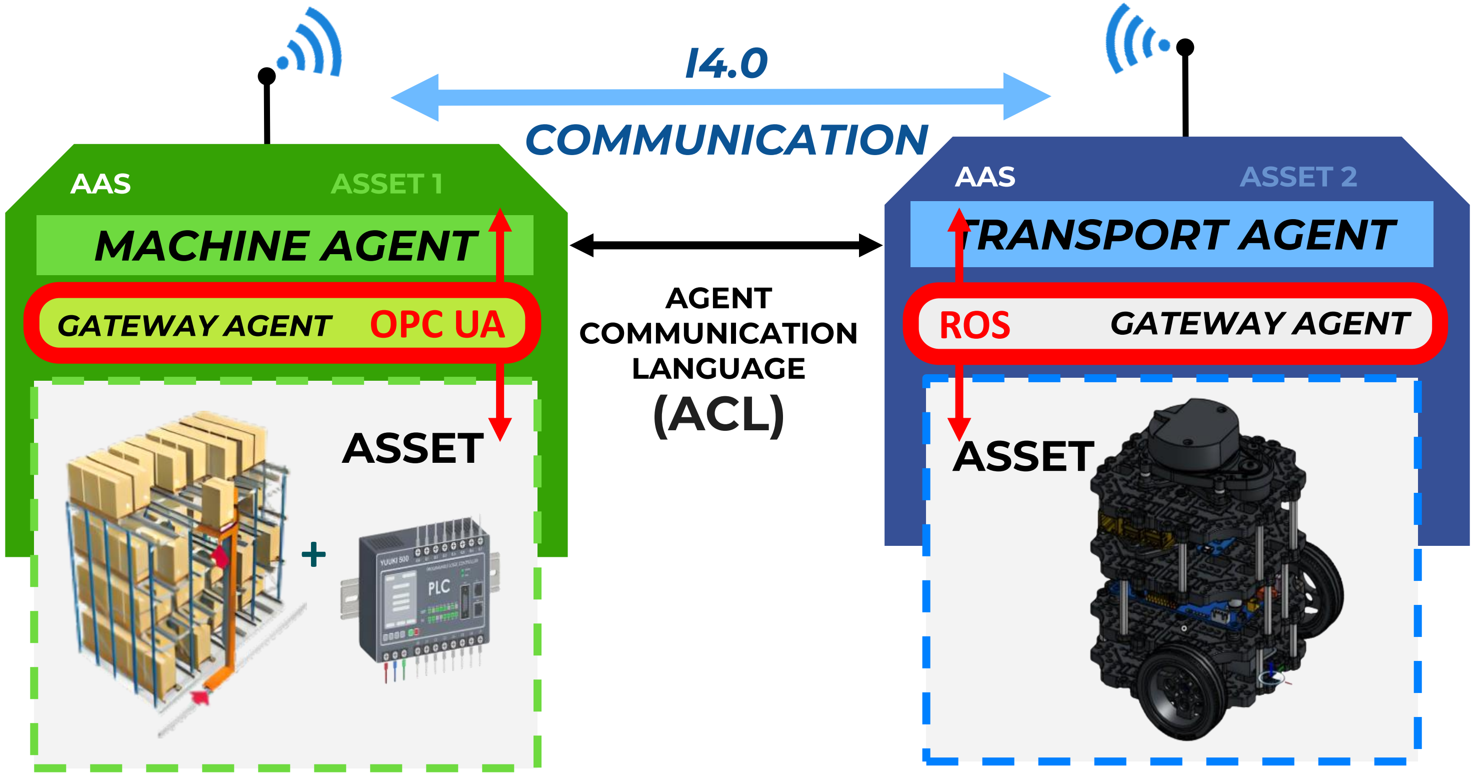
COMPLIANCE WITH IEEE 2660.1 SCENARIOS

GATEWAY DESIGN PATTERN

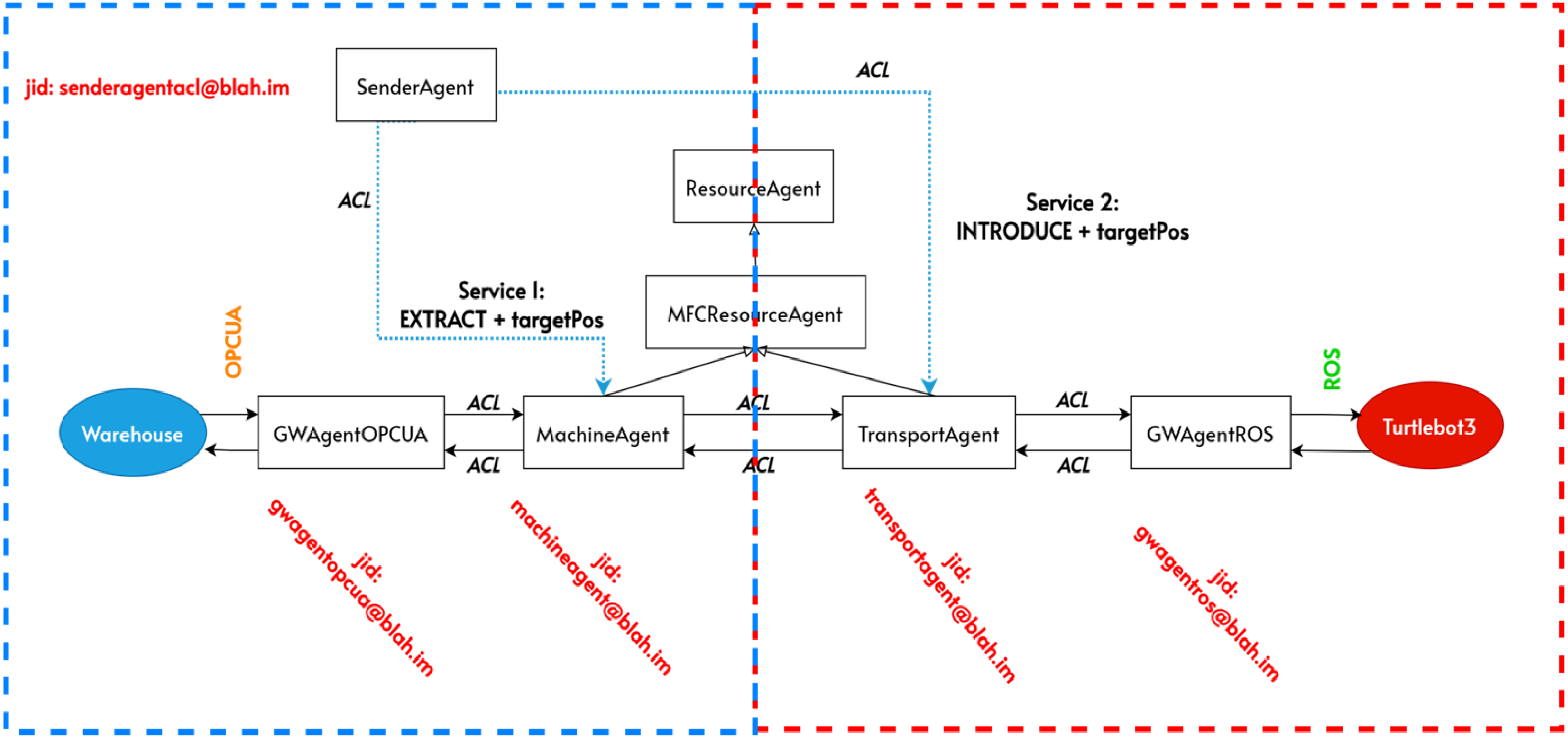


USE CASE

OVERVIEW

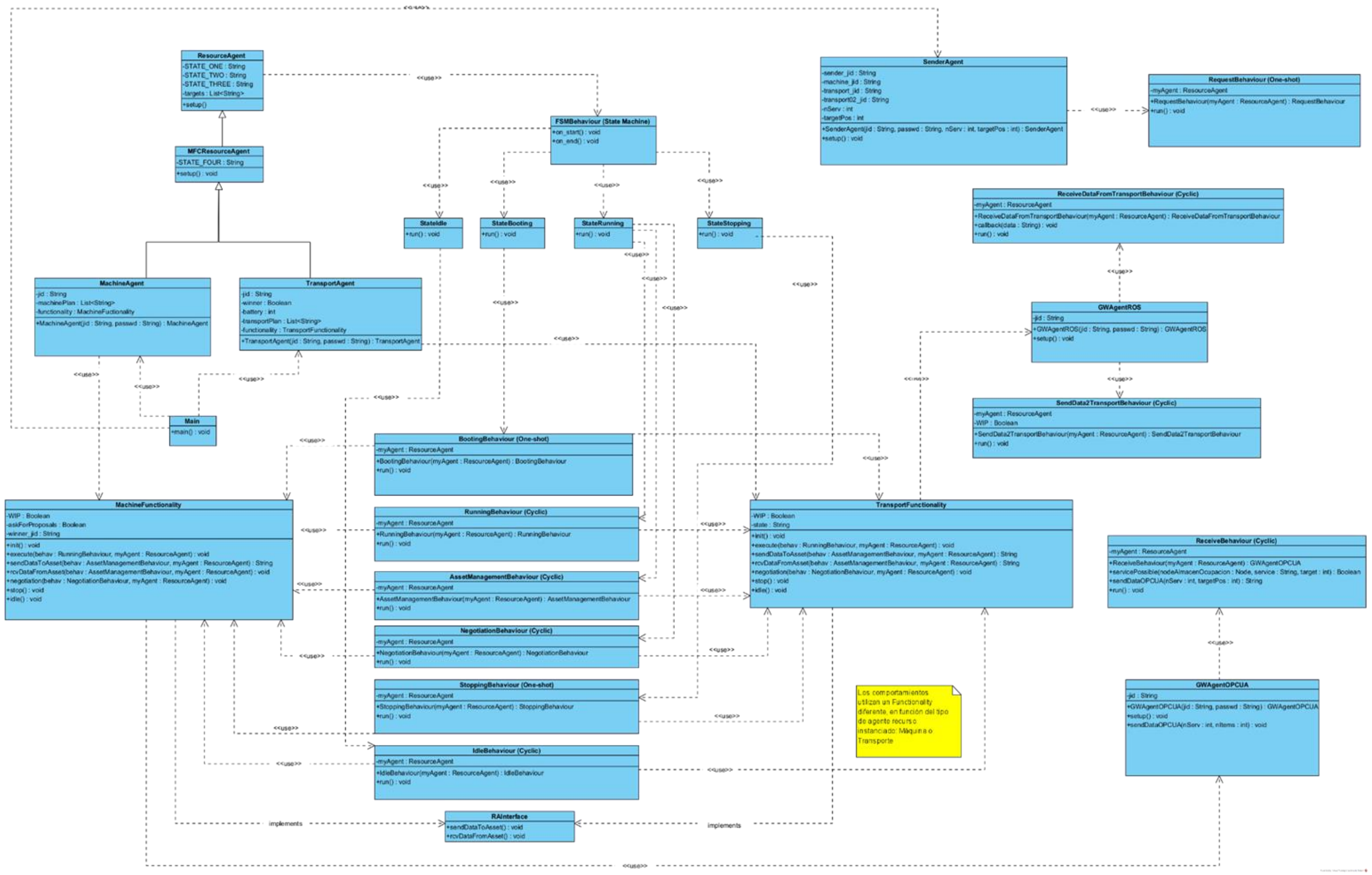


AGENT-BASED SYSTEM STRUCTURE



UML DIAGRAM OF THE PROJECT

- Using MAS does not mean that the solution is simple, but structured.



ResourceAgent
-STATE_ONE : String
-STATE_TWO : String
-STATE_THREE : String
-targets : List<String>
+setup()

MFCResourceAgent
-STATE_FOUR : String
+setup() : void

MachineAgent
-jid : String
-machinePlan : List<String>
-functionality : MachineFuctionality
+MachineAgent(jid : String, passwd : String) : MachineAgent

TransportAgent
-jid : String
-winner : Boolean
-battery : int
-transportPlan : List<String>
-functionality : TransportFunctionality
+TransportAgent(jid : String, passwd : String) : TransportAgent

Main
+main() : void

FSMBehaviour (State Machine)
+on_start() : void
+on_end() : void

StateIdle
+run() : void

StateBooting
+run() : void

StateRunning
+run() : void

StateStopping
+run() : void

SenderAgent
-sender_jid : String
-machine_jid : String
-transport_jid : String
-transport02_jid : String
-nServ : int
-targetPos : int
+SenderAgent(jid : String, passwd : String, nServ : int, targetPos : int) : SenderAgent
+setup() : void

RequestBehaviour (One-shot)
-myAgent : ResourceAgent
+RequestBehaviour(myAgent : ResourceAgent) : RequestBehaviour
+run() : void

ReceiveDataFromTransportBehaviour (Cyclic)
-myAgent : ResourceAgent
+ReceiveDataFromTransportBehaviour(myAgent : ResourceAgent) : ReceiveDataFromTransportBehaviour
+callback(data : String) : void
+run() : void

GWAgentROS
-jid : String
+GWAgentROS(jid : String, passwd : String) : GWAgentROS
+setup() : void

SendData2TransportBehaviour (Cyclic)
-myAgent : ResourceAgent
-WIP : Boolean
+SendData2TransportBehaviour(myAgent : ResourceAgent) : SendData2TransportBehaviour
+run() : void

BootingBehaviour (One-shot)
-myAgent : ResourceAgent
+BootingBehaviour(myAgent : ResourceAgent) : BootingBehaviour
+run() : void

MachineFunctionality
-WIP : Boolean
-askForProposals : Boolean
-winner_jid : String
+init() : void
+execute(behav : RunningBehaviour, myAgent : ResourceAgent) : void
+sendDataToAsset(behav : AssetManagementBehaviour, myAgent : ResourceAgent) : String
+rcvDataFromAsset(behav : AssetManagementBehaviour, myAgent : ResourceAgent) : void
+negotiation(behav : NegotiationBehaviour, myAgent : ResourceAgent) : void
+stop() : void
+idle() : void

RunningBehaviour (Cyclic)
-myAgent : ResourceAgent
+RunningBehaviour(myAgent : ResourceAgent) : RunningBehaviour
+run() : void

AssetManagementBehaviour (Cyclic)
-myAgent : ResourceAgent
+AssetManagementBehaviour(myAgent : ResourceAgent) : AssetManagementBehaviour
+run() : void

NegotiationBehaviour (Cyclic)
-myAgent : ResourceAgent
+NegotiationBehaviour(myAgent : ResourceAgent) : NegotiationBehaviour
+run() : void

StoppingBehaviour (One-shot)
-myAgent : ResourceAgent
+StoppingBehaviour(myAgent : ResourceAgent) : StoppingBehaviour
+run() : void

IdleBehaviour (Cyclic)
-myAgent : ResourceAgent
+IdleBehaviour(myAgent : ResourceAgent) : IdleBehaviour
+run() : void

RAInterface
+sendDataToAsset() : void
+rcvDataFromAsset() : void

TransportFunctionality
-WIP : Boolean
-state : String
+init() : void
+execute(behav : RunningBehaviour, myAgent : ResourceAgent) : void
+sendDataToAsset(behav : AssetManagementBehaviour, myAgent : ResourceAgent) : String
+rcvDataFromAsset(behav : AssetManagementBehaviour, myAgent : ResourceAgent) : String
+negotiation(behav : NegotiationBehaviour, myAgent : ResourceAgent) : void
+stop() : void
+idle() : void

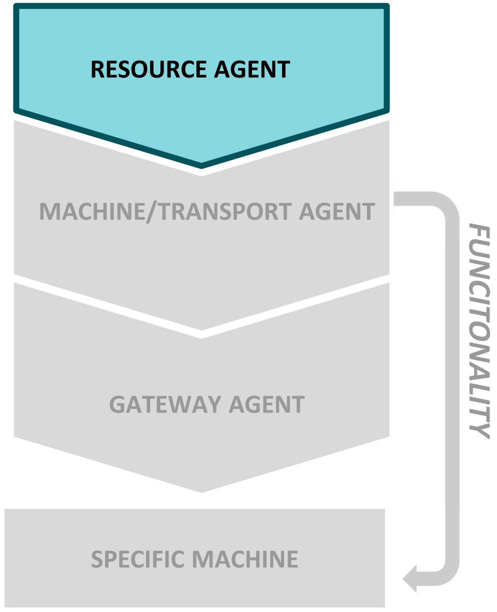
ReceiveBehaviour (Cyclic)
-myAgent : ResourceAgent
+ReceiveBehaviour(myAgent : ResourceAgent) : GWAgentOPCUA
+servicePossible(nodeAlmacenOcupacion : Node, service : String, target : int) : Boolean
+sendDataOPCUA(nServ : int, targetPos : int) : String
+run() : void

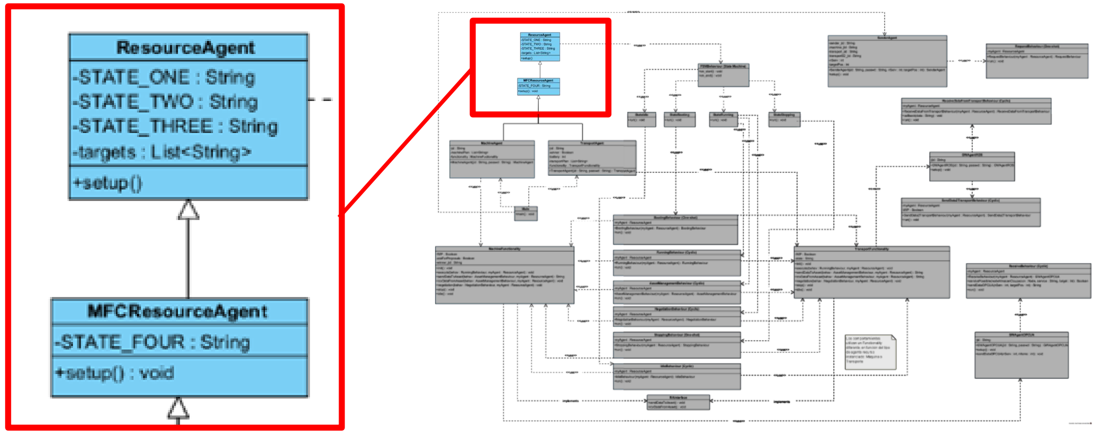
GWAgentOPCUA
-jid : String
+GWAgentOPCUA(jid : String, passwd : String) : GWAgentOPCUA
+setup() : void
+sendDataOPCUA(nServ : int, nItems : int) : void

Los comportamientos utilizan un Functionality diferente, en función del tipo de agento recurso instanciado: Máquina o Transporte

UML DIAGRAM OF THE PROJECT

- Using MAS does not mean that the solution is simple, but structured.



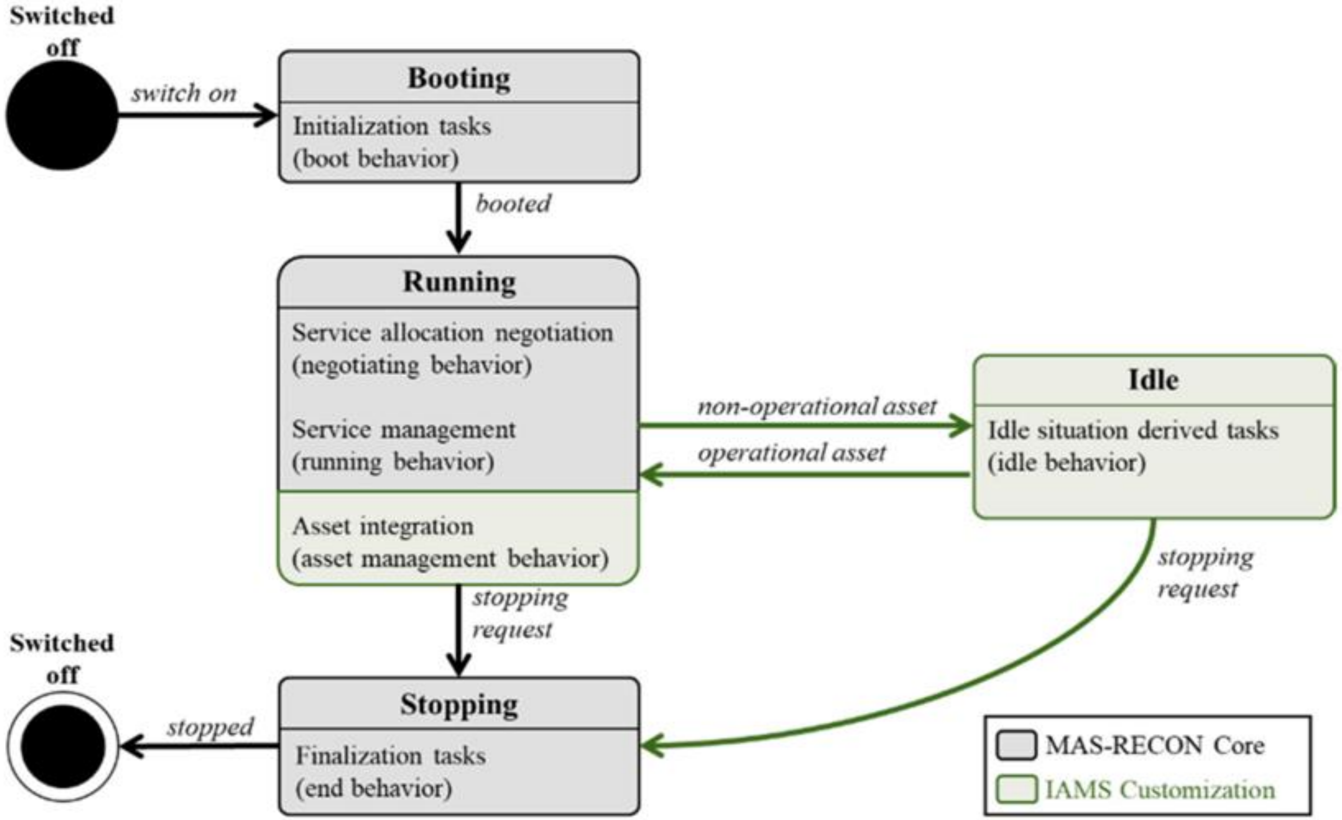


ResourceAgent | MFCResourceAgent

- ♦ Any type of resource, generic
- ♦ Abstract and non-instantiable class → Inheritance from other classes.
- ♦ FSM behavior: Booting, Running and Stopping.
- ♦ Manufacturing ResourceAgent: 4th state to FSM → IDLE or idle.

FINITE STATE MACHINE FOR RESOURCE AGENTS

- All agents have common behaviours
- All these behaviors are divided into a set of common states through which all agents pass.



Switched off

switch on

Booting

Initialization tasks
(boot behavior)

booted

Running

Service allocation negotiation
(negotiating behavior)

Service management
(running behavior)

Asset integration
(asset management behavior)

non-operational asset

operational asset

Idle

Idle situation derived tasks
(idle behavior)

stopping request

stopping request

Stopping

Finalization tasks
(end behavior)

stopped

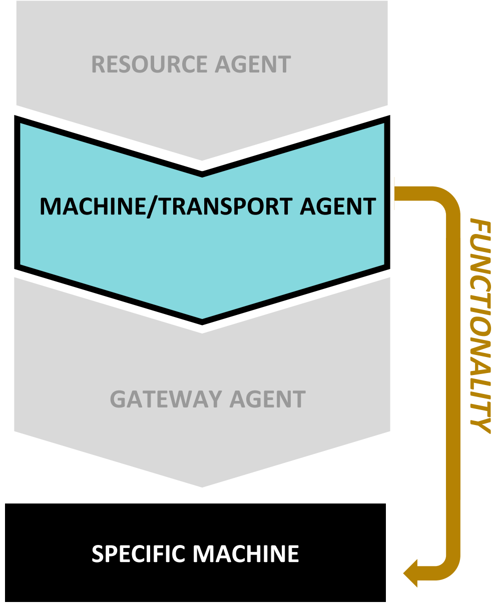
Switched off

□ MAS-RECON Core

🟩 IAMS Customization

UML DIAGRAM OF THE PROJECT

- Using MAS does not mean that the solution is simple, but structured.



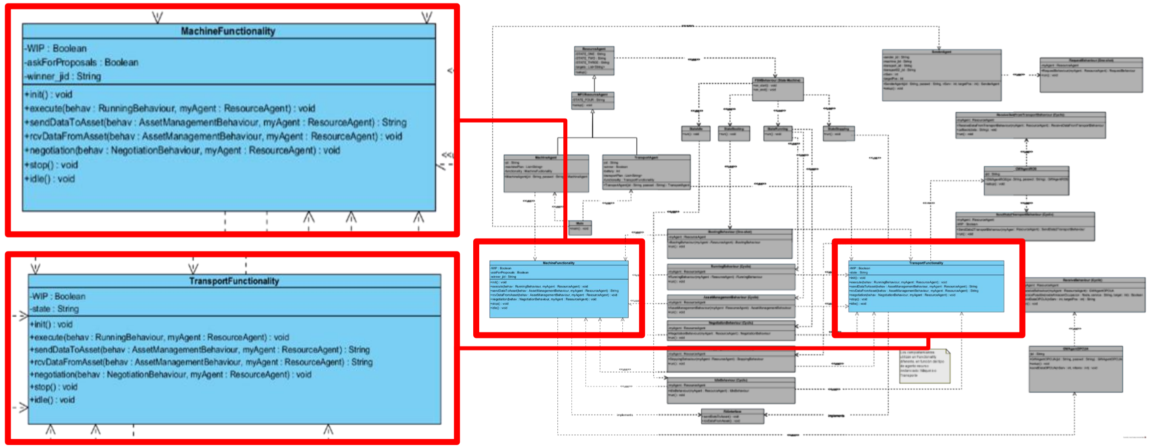
RESOURCE AGENT

MACHINE/TRANSPORT AGENT

GATEWAY AGENT

SPECIFIC MACHINE

FUNCTIONALITY

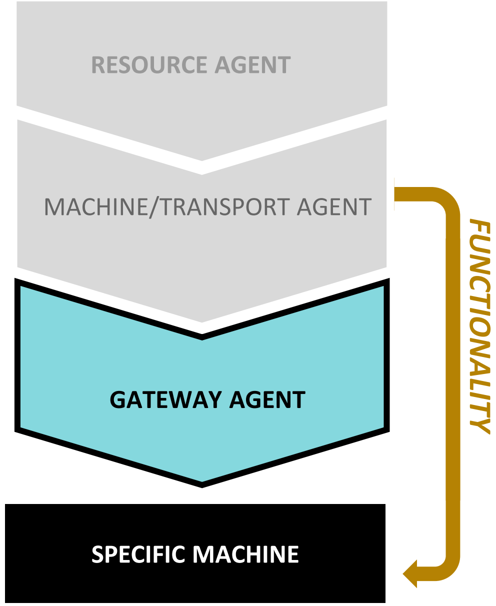

MachineFunctionality
-WIP : Boolean
-askForProposals : Boolean
-winner_jid : String
+init() : void
+execute(behav : RunningBehaviour, myAgent : ResourceAgent) : void
+sendDataToAsset(behav : AssetManagementBehaviour, myAgent : ResourceAgent) : String
+rcvDataFromAsset(behav : AssetManagementBehaviour, myAgent : ResourceAgent) : void
+negotiation(behav : NegotiationBehaviour, myAgent : ResourceAgent) : void
+stop() : void
+idle() : void
TransportFunctionality
-WIP : Boolean
-state : String
+init() : void
+execute(behav : RunningBehaviour, myAgent : ResourceAgent) : void
+sendDataToAsset(behav : AssetManagementBehaviour, myAgent : ResourceAgent) : String
+rcvDataFromAsset(behav : AssetManagementBehaviour, myAgent : ResourceAgent) : String
+negotiation(behav : NegotiationBehaviour, myAgent : ResourceAgent) : void
+stop() : void
+idle() : void

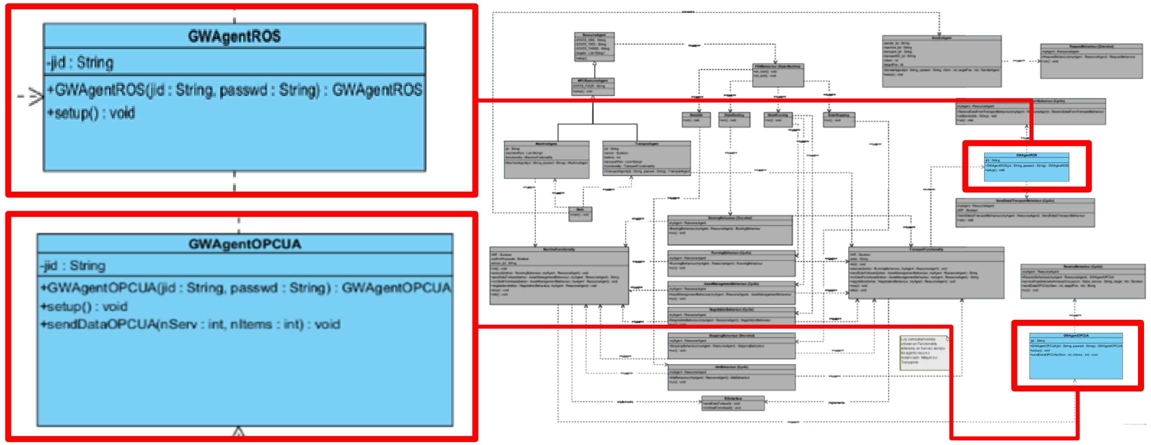
MachineFunctionality | TransportFunctionality

- ♦ Specific features of concrete device are concentrated in this class.
- ♦ **This allows using the same agent template for all the assets in the factory.**
- ♦ **Thus, the end user only has to customize the agent using this class (i.e., they create design patterns from the skeleton pattern)**

UML DIAGRAM OF THE PROJECT

- Using MAS does not mean that the solution is simple, but structured.

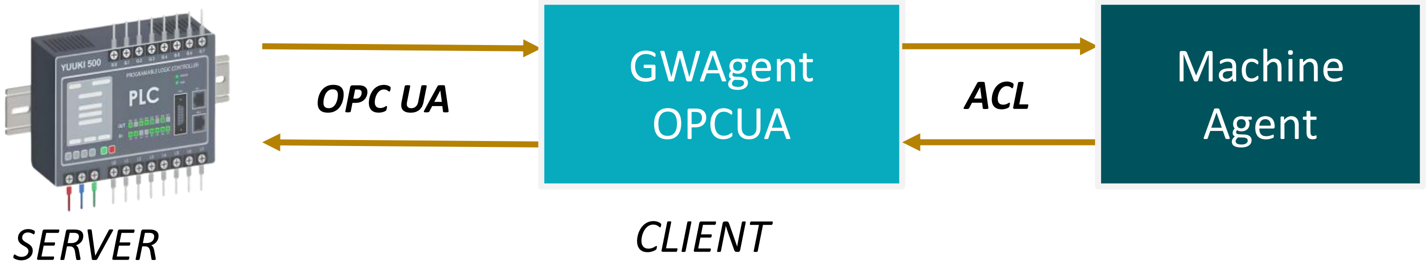


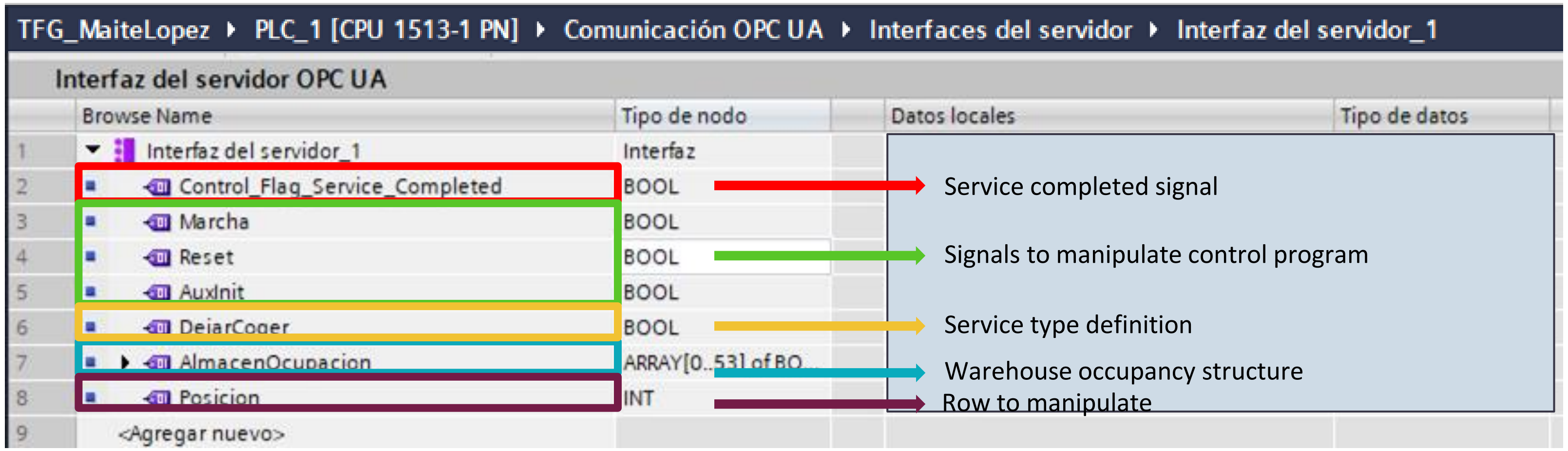

GWAgentROS
-jid : String
+GWAgentROS(jid : String, passwd : String) : GWAgentROS
+setup() : void
GWAgentOPCUA
-jid : String
+GWAgentOPCUA(jid : String, passwd : String) : GWAgentOPCUA
+setup() : void
+sendDataOPCUA(nServ : int, nItems : int) : void

GWAgentOPCUA | GWAgentROS

- ♦ Independence of communication particularities
- ♦ Machine using an OPC UA interface (client-server)
- ♦ Translate messages: agent (ACL) ↔ active (OPCUA/ROS).
  - ♦ Duality

AGENT SYSTEM STRUCTURE: MACHINE AGENT




TFG_MaiteLopez ▸ PLC_1 [CPU 1513-1 PN] ▸ Comunicación OPC UA ▸ Interfaces del servidor ▸ Interfaz del servidor_1

Interfaz del servidor OPC UA

	Browse Name	Tipo de nodo	Datos locales	Tipo de datos
1	▾ Interfaz del servidor_1	Interfaz		
2	Control_Flag_Service_Completed	BOOL		
3	Marcha	BOOL		
4	Reset	BOOL		
5	AuxInit	BOOL		
6	DejarCoger	BOOL		
7	▸ AlmacenOcupacion	ARRAY[0..53] of BO...		
8	Posicion	INT		
9	<Agregar nuevo>			

Service completed signal

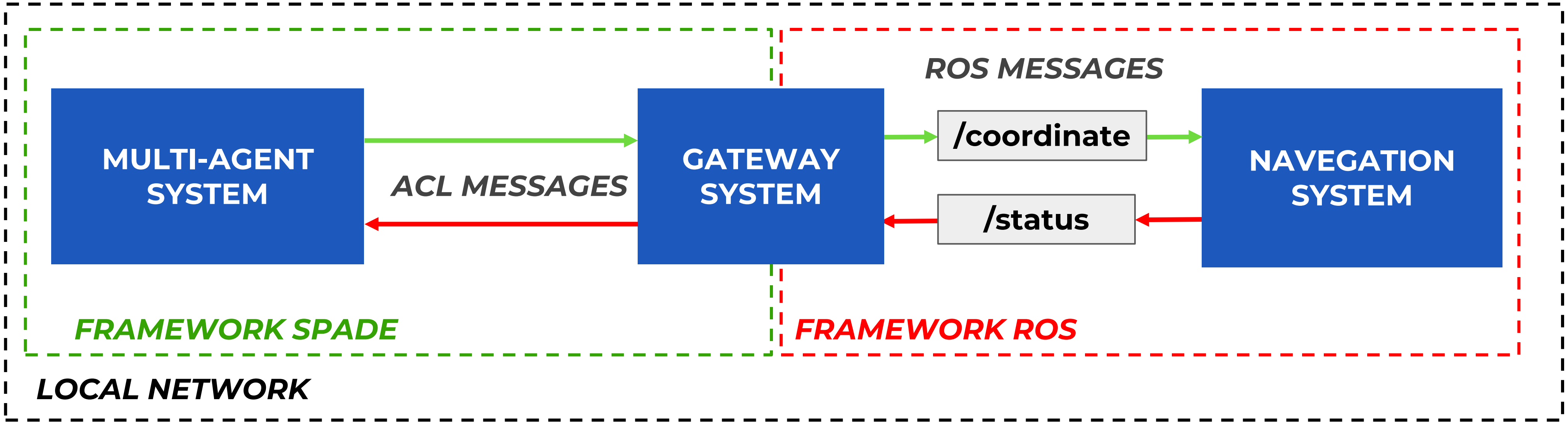
Signals to manipulate control program

Service type definition

Warehouse occupancy structure

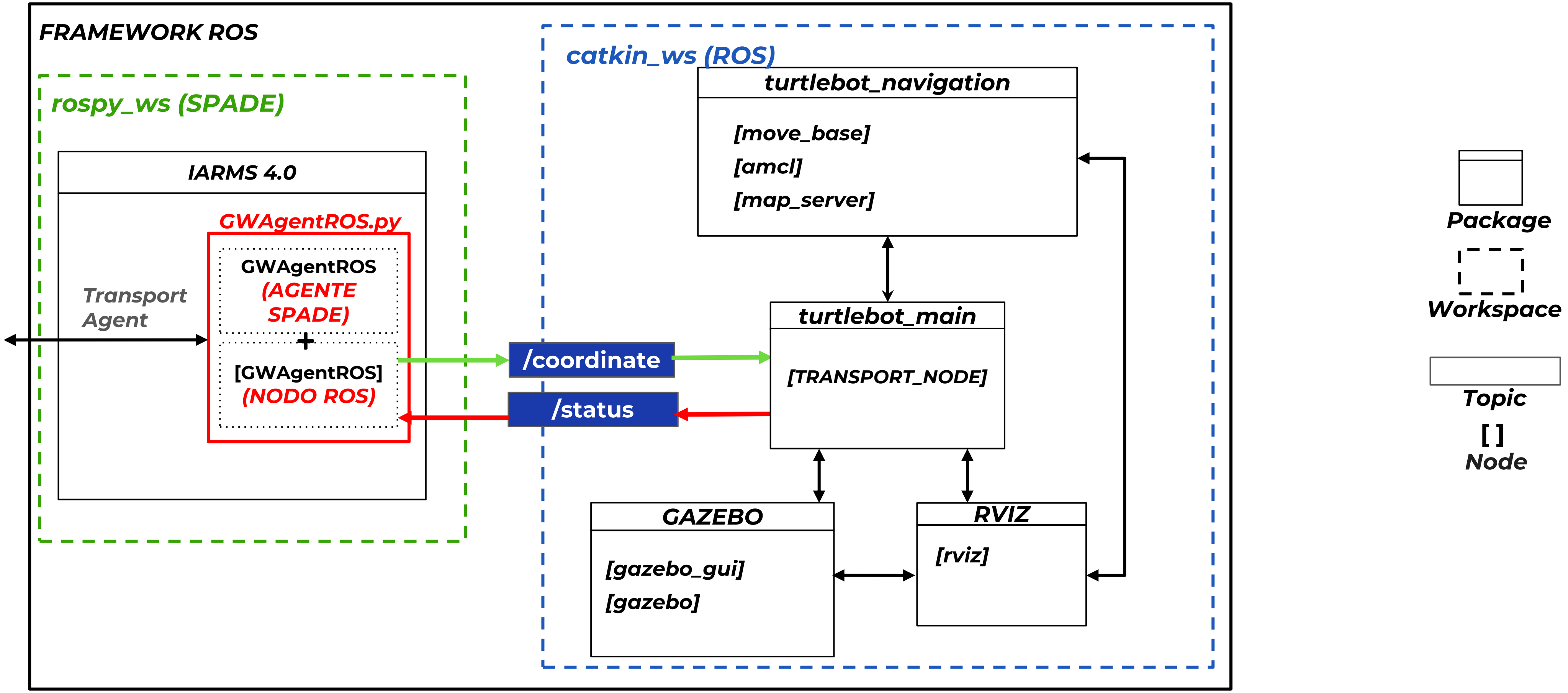
Row to manipulate

AGENT SYSTEM STRUCTURE: TRANSPORT AGENT



- ***/coordinate:*** Transport services requests
- ***/status:*** Read the status of the robot

AGENT SYSTEM STRUCTURE: TRANSPORT AGENT


FRAMEWORK ROS
rospy_ws (SPADE)
IARMS 4.0
GWAgentROS.py
Transport Agent
GWAgentROS
(AGENTE SPADE)
+
[GWAgentROS]
(NODO ROS)
/coordinate
/status
catkin_ws (ROS)
turtlebot_navigation
[move_base]
[amcl]
[map_server]
turtlebot_main
[TRANSPORT_NODE]
GAZEBO
[gazebo_gui]
[gazebo]
RVIZ
[rviz]
Package
Workspace
Topic
[ ]
Node