FACTORY

Agents and Distributed Artificial Intelligence - November 2019
Integrated Masters in Informatics and Computer Engineering @ FEUP

Group 13:

Duarte Faria	up201607176
Mariana Aguiar	up201605904
Tiago Fragoso	up201606040

PROBLEM DESCRIPTION

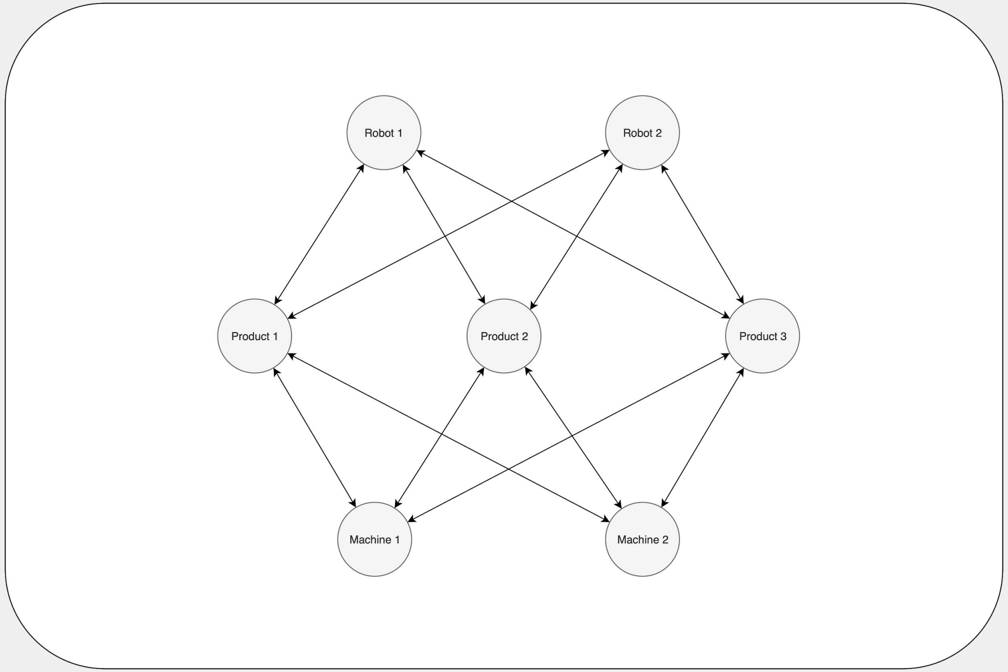
A factory produces several different **products**, composed by a series of **processes** that need to be completed in a specific predefined order.
There are several different **machines**, each one with a list of the different **processes** that it can perform and an associated duration time to each process. The time a process takes to be completed can vary from one machine to another.

All products start at a predefined **pick up** point and need a **robot** to take them from that point to a machine and between different machines and to the predefined drop off point once the product is complete.
A **robot** can transport every type of products in between any two points. It also has a velocity at which it travels a space unit.

Products, robots and machines should communicate and interact in order to schedule which processes of which products each machine will perform and which products a robot will transport, in order to minimize the total amount of time spent to complete each product.

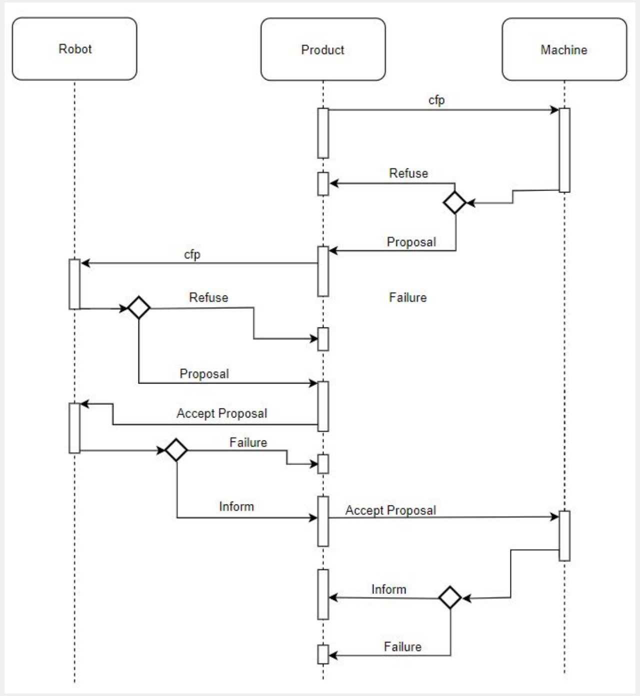
AGENTS

For this project were implemented three kinds of agents: products, machines, robots. Their basic interactions can be seen in the following diagram:



INTERACTIONS & PROTOCOLS

The communications in this project follow the communication protocol described in the following diagram:



INTERACTIONS AND PROTOCOLS

The products initiate a **Contract Network Protocol**, as seen in the diagram, for each one of its processes, in order to schedule that process in the machine with the best proposal.
A product starts out by sending a **CFP** message, with the process code it wants to complete, to all the machines available in the system.
Each machine checks if it can perform that process and if so sends a **PROPOSE** message with a start time, the first instant the machine is available and a duration time, If it cannot perform that process, it sends a **REFUSE** message.
Upon receiving all the proposals from the machines, the product needs to find a robot to transport him to a machine that performs the process. To do this the product sends a **CFP** message with the pickup point (its current position) and drop off point (machine position) for each received machine proposal to all robots available.
Each available robot sends a **PROPOSE** message with the time he is available and the duration time of the displacement, the pick up and the drop off point.
Upon receiving all the journey proposals form the robots, the product chooses the best proposal, the one that, joining the robot journey with the machine tasks provides the smaller end time and sends a **ACCEPT_PROPOSAL** message to the corresponding robot.

INTERACTIONS AND PROTOCOLS

The robot receives the **ACCEPT_PROPOSAL** message sent by the product and checks if in the meantime it didn't schedule another journey, and if so it sends a **FAILURE** message. Otherwise it sends an **INFORM** message. Afterwards, the product sends an **ACCEPT_PRPOPOSAL** message to the machine for which the journey was scheduled. If the product is not available before the start time sent by the machine, it includes a new time in the **ACCEPT_PROPOSAL** message.
The machine, after receiving the **ACCEPT_PROPOSAL** message checks in the meantime it didn't schedule another process in that time interval and sends an **INFORM** message if it is available and a **FAILURE** message, otherwise.
If the product receives a **FAILURE** message from the machine, it sends a **CANCEL** message to the robot for the previously scheduled journey and starts the communication protocol again for the same process.

OTHER INTERACTIONS (AGENT DISCOVERY)

A **Directory Facilitator** (DF) agent was used for the agent discovery interactions.

The products are the only agents to initiate communications, due to this all agent discovery actions were implemented in a product agent behaviour, **ProductBehaviour**.

This behaviour uses the DFService to search for all the agents of a certain type, for example to search for the machine agents, it looks for the agents with the type **“machine”** and to look for the robot agents it uses the type **“robot”**.

Upon receiving the results for the agent search, these are stored in a data structure that is later used when waiting for the responses to the proposal requests to verify that all the agents have answered.

EXPERIMENTS

The experiments were performed by varying the complexity of each product (number of processes), the versatility of machines and the number of robots, as well as their speed.
The experiments can be defined in the examples folder by the means of a text file specifying the environment, which can then be run as the factory floor plan.

Some of the run experiments include:

- 1 Product, 1 Machine, 1 Robot (simple test)
- 2 Products, 2 Machines, 1 Robot **vs** 2 Products, 2 Machines, 2 Robots (robot increase)
- 7 Products, 2 Machines, 4 Robots **vs** 7 Products, 4 Machines, 4 Robots (machine increase)
- 21 Products, 4 Machines, 4 Robots

At the end of the experiment, the maximum scheduling time is printed in the console and detailed schedule tables are logged into a file. The logs from the negotiations can be observed while the program is running.

RESULTS ANALYSIS

From the limited experiments performed, some conclusions can be drawn:

- The system is scalable (the increase in agents does not seem to interfere with the quality of the solution)
- The solution is not optimal (expected since the scheduling happens on a forward only scale, thus not allowing for downtime to be minimized)
- The bottleneck is the number of machines (and their versatility/speed)
- The number of robots does not seem to have great impact on the solution unless their speed is very low
- Due to the asynchronous nature of the communication, the greater the number of agents, the more divergent the solutions can become for the same problem.

CONCLUSION

All in all, the implementation is considered to have been successful, being able to accomplish what we set out to do. Unfortunately, we were not able to introduce the product priority concept into the project because we decided to increase the complexity by adding a new Agent (Robot).

We are looking forward to performing more thorough and intense experiments in the system, so that we can analyze the results and extrapolate more concrete data about the quality of the implementation and solutions.

ADDITIONAL INFORMATION

Execution and implementation details

DETAILED EXECUTION EXAMPLES

The program can be run by running the following command inside the ***scripts*** folder:

sh run.sh <filename> [--verbose]

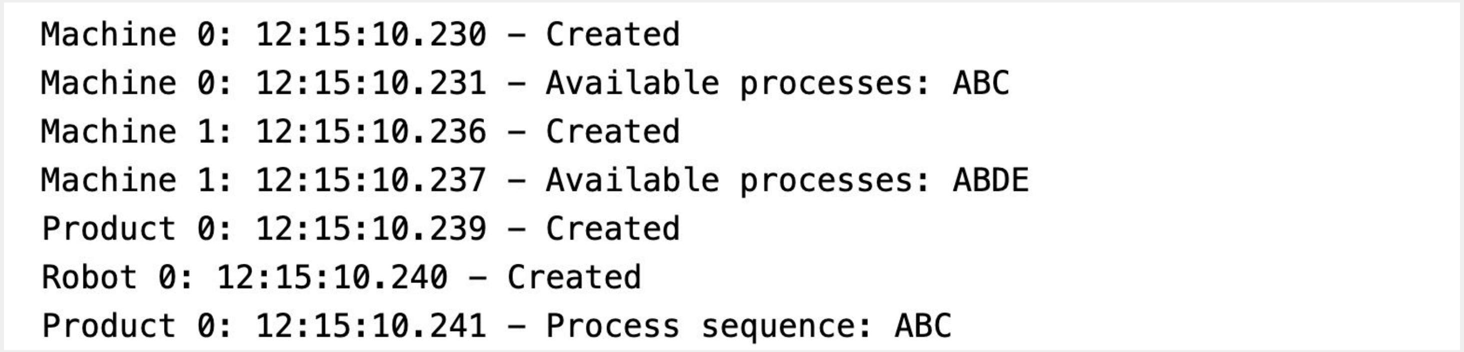
The *filename* refers to the text file specifying the environment, which should be located under the **examples** folder.

The *verbose* flag enables more extensive logging during the negotiations.

The schedule tables can be consulted under the ***logs*** folder;

DETAILED EXECUTION EXAMPLES

After initiating the program with the following global variables, the logs show the creation of the agents.



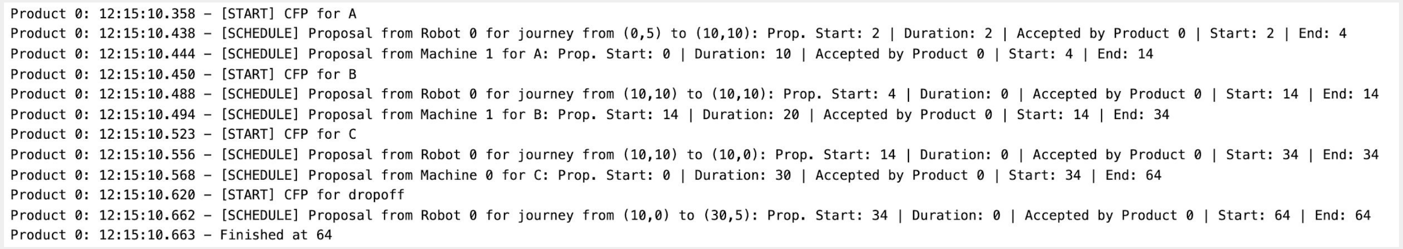
```
Machine 0: 12:15:10.230 – Created
Machine 0: 12:15:10.231 – Available processes: ABC
Machine 1: 12:15:10.236 – Created
Machine 1: 12:15:10.237 – Available processes: ABDE
Product 0: 12:15:10.239 – Created
Robot 0: 12:15:10.240 – Created
Product 0: 12:15:10.241 – Process sequence: ABC
```

DETAILED EXECUTION EXAMPLES

After the creation of the agents, the logs show the product send a CFP message to initiate the protocol and schedule the necessary journeys and tasks.

The logs show every robot journey and machine task scheduled and their details.

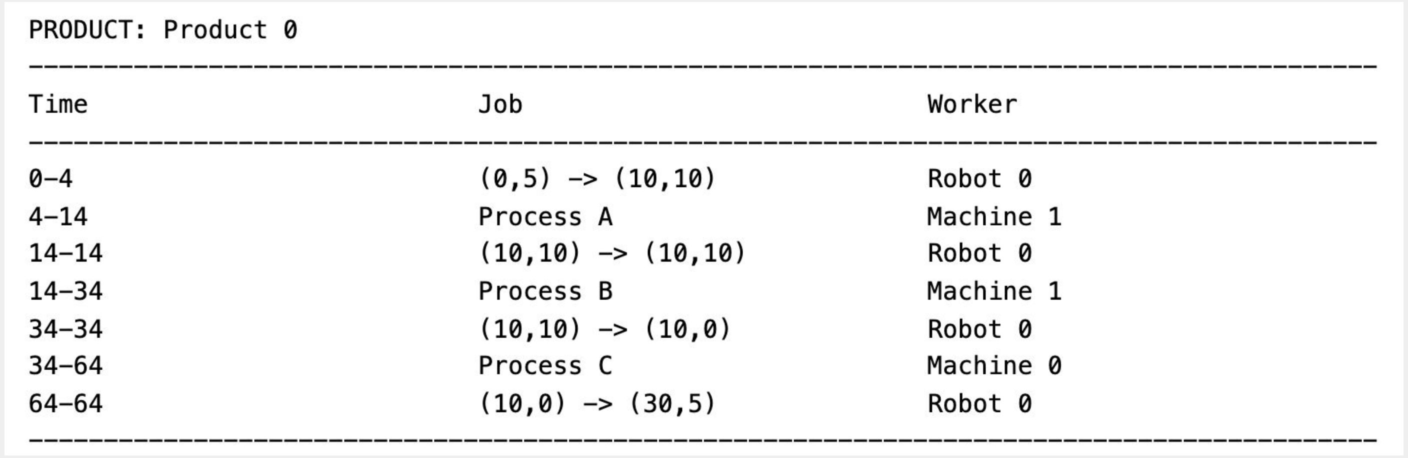
(using the --*verbose* flag)



```
Product 0: 12:15:10.358 - [START] CFP for A
Product 0: 12:15:10.438 - [SCHEDULE] Proposal from Robot 0 for journey from (0,5) to (10,10): Prop. Start: 2 | Duration: 2 | Accepted by Product 0 | Start: 2 | End: 4
Product 0: 12:15:10.444 - [SCHEDULE] Proposal from Machine 1 for A: Prop. Start: 0 | Duration: 10 | Accepted by Product 0 | Start: 4 | End: 14
Product 0: 12:15:10.450 - [START] CFP for B
Product 0: 12:15:10.488 - [SCHEDULE] Proposal from Robot 0 for journey from (10,10) to (10,10): Prop. Start: 4 | Duration: 0 | Accepted by Product 0 | Start: 14 | End: 14
Product 0: 12:15:10.494 - [SCHEDULE] Proposal from Machine 1 for B: Prop. Start: 14 | Duration: 20 | Accepted by Product 0 | Start: 14 | End: 34
Product 0: 12:15:10.523 - [START] CFP for C
Product 0: 12:15:10.556 - [SCHEDULE] Proposal from Robot 0 for journey from (10,10) to (10,0): Prop. Start: 14 | Duration: 0 | Accepted by Product 0 | Start: 34 | End: 34
Product 0: 12:15:10.568 - [SCHEDULE] Proposal from Machine 0 for C: Prop. Start: 0 | Duration: 30 | Accepted by Product 0 | Start: 34 | End: 64
Product 0: 12:15:10.620 - [START] CFP for dropoff
Product 0: 12:15:10.662 - [SCHEDULE] Proposal from Robot 0 for journey from (10,0) to (30,5): Prop. Start: 34 | Duration: 0 | Accepted by Product 0 | Start: 64 | End: 64
Product 0: 12:15:10.663 - Finished at 64
```

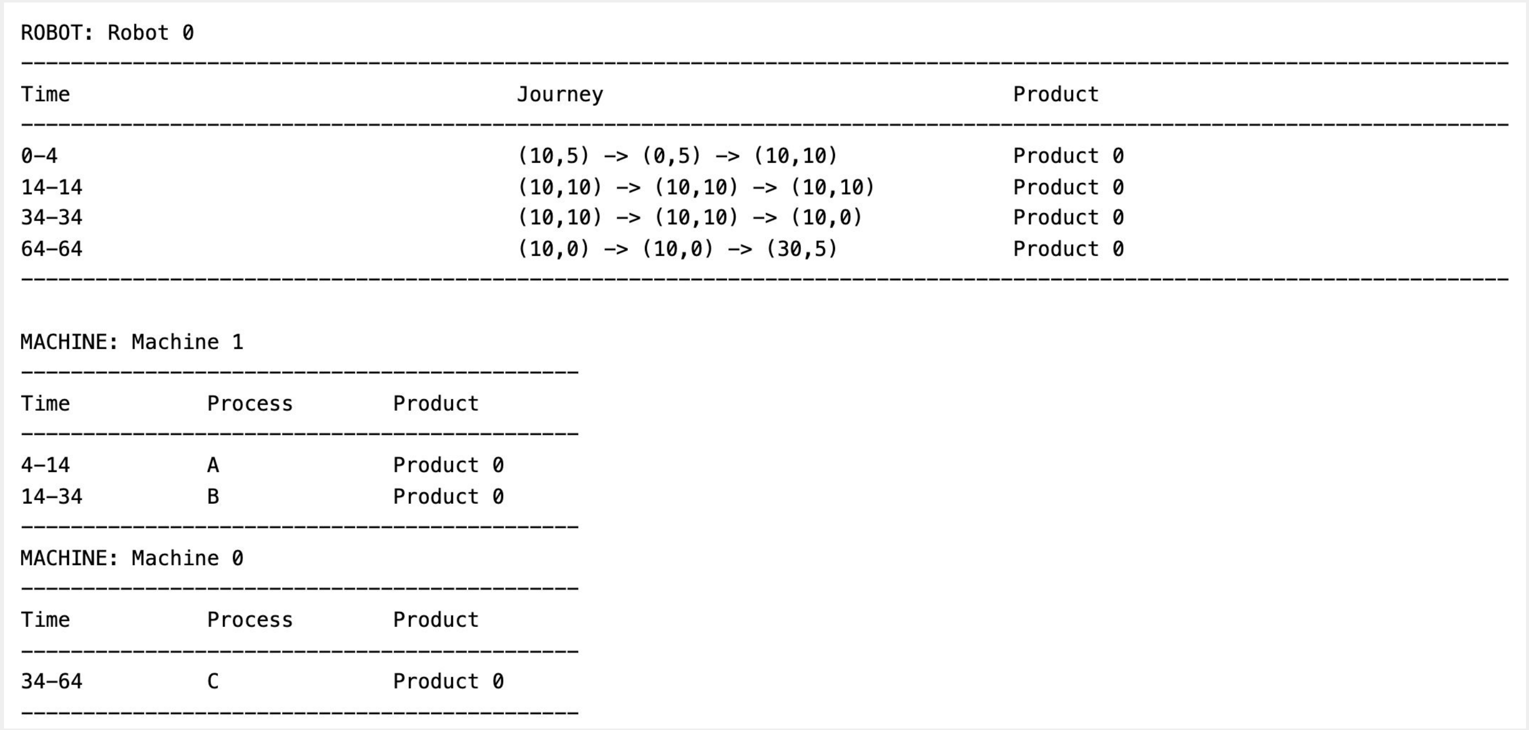
DETAILED EXECUTION EXAMPLES

After all the products have concluded their scheduling, the schedule of every agent in the system can be consulted.



```
PRODUCT: Product 0
------------------------------------------------------------------------------------------
Time                          Job                           Worker
------------------------------------------------------------------------------------------
0-4                           (0,5) -> (10,10)              Robot 0
4-14                          Process A                     Machine 1
14-14                         (10,10) -> (10,10)            Robot 0
14-34                         Process B                     Machine 1
34-34                         (10,10) -> (10,0)             Robot 0
34-64                         Process C                     Machine 0
64-64                         (10,0) -> (30,5)              Robot 0
------------------------------------------------------------------------------------------
```

DETAILED EXECUTION EXAMPLES



```
ROBOT: Robot 0
------------------------------------------------------------------------------------------------------------------------
Time                                    Journey                                 Product
------------------------------------------------------------------------------------------------------------------------
0-4                                     (10,5) -> (0,5) -> (10,10)              Product 0
14-14                                   (10,10) -> (10,10) -> (10,10)           Product 0
34-34                                   (10,10) -> (10,10) -> (10,0)            Product 0
64-64                                   (10,0) -> (10,0) -> (30,5)              Product 0
------------------------------------------------------------------------------------------------------------------------


MACHINE: Machine 1
---------------------------------------------
Time           Process        Product
---------------------------------------------
4-14           A              Product 0
14-34          B              Product 0
---------------------------------------------
MACHINE: Machine 0
---------------------------------------------
Time           Process        Product
---------------------------------------------
34-64          C              Product 0
---------------------------------------------
```

IMPLEMENTED CLASSES

The classes implemented in the development of this project were the following:

- Agent classes:
  - **ProductAgent**: represents a product in the factory and is responsible for starting the communication protocol
  - **MachineAgent**: represents a machine in the factory
  - **RobotAgent**: represents a robot in the factory

- Auxiliary classes:
  - **Process**: represents a product process and has an unique code
  - **Job**: represents an action of a working agent (machine or robot) and has a start time, end time, product and worker associated
  - **Task**: represents an action of a machine and has a process and a location
  - **Proposal**: represents a machine proposal for a process and has a process, machine, duration and machine earliest time available
  - **Journey**: represents an action of a robot and has a start point and end point
  - **JourneyProposal**: represents a robot proposal for a journey and has a pick up point, drop off point, pick up duration, duration and robot earliest time available

IMPLEMENTED CLASSES: PRODUCT AGENT

This class, ProductAgent, represents a product in the factory and is responsible for starting the communications protocols. The class has the following attributes:

- **processes**: *LinkedHashMap* with the list of processes for the product and a *boolean* to indicate if a process was already scheduled or not.
- **scheduledJobs**: *ArrayList* with all the schedules Jobs (tasks and journeys) for that product, works as the product schedule.
- **startingPoint**: *Point* at which the product starts at, the same as the Factory's pick up point.

And the following behaviours:

- **ProductBehaviour**: This behaviour is responsible for searching for all the machine and robot agents in the system and for starting a new **ScheduleProcess** behaviour for the process list.
- **ScheduleProcess**: This behaviour handles all the communication for the product, makes the choices for the best proposals and starts the communication protocol described in the Interaction and Protocols section.

IMPLEMENTED CLASSES: MACHINE AGENT

This class, **MachineAgent**, represents a machine in the factory and has the following attributes:

- **availableProcess**: *HashMap* with each process, that the machine can perform, and its corresponding duration time.
- **scheduledTasks**: *ArrayList* with all the Tasks already scheduled in that machine
- **location**: *Point* at which the machine is located in the factory.

And the following behaviours:

- **ReplyToRequestBehaviour**: This behaviour receives all the process requests sent by the products and responds with a *Proposal*, if it can perform that process , that contains a duration, location of the machine, process and the machine's earliest available time.
- **ScheduleTaskBehaviour**: This behaviour handles all the ACCEPT_PROPOSAL messages sent by the products that are composed by a *Proposal* similar to the one sent by the product but with a product start time, that corresponds to the product's earliest available time. The behaviour then checks if the machine is still available at that time and responds with an INFORM or with a FAILURE.

IMPLEMENTED CLASSES: ROBOT AGENT

This class, **RobotAgent**, represents a robot in the factory and has the following attributes:

- **velocity**: velocity at which the robot travels one space unit.
- **scheduledJourneys**: *ArrayList* with all the Journeys already scheduled in that robot.
- **startingPoint**: *Point* at which the robot starts at.

And the following behaviours:

- **RobotReplyToRequestBehaviour**: This behaviour receives all the journey requests sent by the products and responds with a *JourneyProposal*, that contains a pick up point, drop off point, durantion, pick up duration and the robot's earliest available time.
- **ScheduleJourneyBehaviour**: This behaviour handles all the ACCEPT_PROPOSAL messages sent by the products that are composed by a *JourneyProposal* similar to the one sent by the product but with a product start time, that corresponds to the product's earliest available time. The behaviour then checks if the robot is still available at that time and responds with an INFORM or with a FAILURE.
- **CancelJourneyBehaviour**: This behaviour handles the CANCEL messages sent by the products and removes from the scheduled journeys the journeys received.