LABORATORIO DI INGEGNERIA DEI SISTEMI SOFTWARE

Sprint 0

Introduction

Requirements

Automated Car-Parking

A company intends to build an *automating parking service* composed of a set of elements:

- A software system, named ***ParkManagerService***, that implements the required automation functions.
- A DDR robot working as a ***transport trolley***, that is intially situated in its ***home*** location. The transport trolley has the form of a square of side length ***RD***.
- A ***parking-area*** is an empty room that includes;
  - an ***INDOOR*** to enter the car in the area. Facing the INDOOR, there is a ***INDOOR-area*** equipped with a ***weigthsensor*** that measures the ***weigth*** of the car;
  - an ***OUTDOOR*** to exit from the parking-area. Just after the OUTDOOR, there is ***OUTDOOR-area*** equipped with a ***outsonar***, used to detect the presence of a car. The OUTDOOR-area, once engaded by a car, should be freed within a prefixed interval of time ***DTFREE***;
  - a number N (N=6) of ***parking-slots***;
  - a ***thermometer*** that measures the temperature ***TA*** of the area;
  - a ***fan*** that should be activated when **TA > TMAX**, where ***TMAX*** is a prefixed value (e.g. 35)

  A ***map*** of the parking area, represented as a grid of squares of side length ***RD***, is available in the file parkingMap.txt:

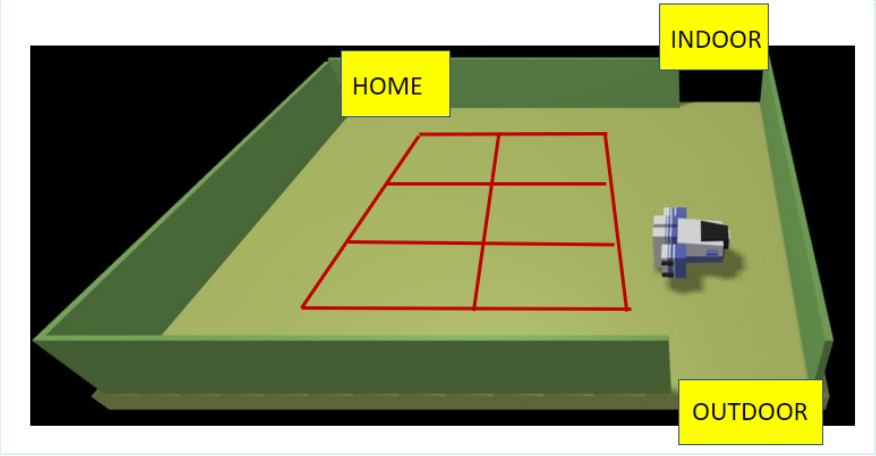
```
|r, 0, 0, 0, 0, 0, 0, X,
|0, 0, X, X,  0, 0, 0, X,
|0, 0, X, X,  0, 0, 0, X,
```

```
|0, 0, X, X,  0, 0, 0, X,
|0, 0, 0, 0, 0, 0, 0, X,
|X, X, X, X, X, X, X, X,
```

The map includes the positions of the parking-slots (marked above with the symbol **X**) and of the ***fixed obstacles*** in the area (the walls marked with the symbol **X**).

The area marked with **X** is a sort of 'equipped area' upon which the transport trolley cannot walk. Thus, to get the car in the parking-slot **(2,2)**, the transport trolley must go in cell **(1,2)**.

The proper scene for the WEnv is reported in: parkingAreaConfig.js



- a ***parking-manager*** (an human being) which supervises the state of the parking-area and handles critical situations.

The job of our company is to design, build and deploy the ***ParkManagerService***.

User stories

As a **client - parking phase** :

- I intend to use a ***ParkServiceGUI*** provided by the ParkManagerService to notify my interest in *entering* my auto in the parking-area and to receive as answer the number ***SLOTNUM*** of a free parking-slot (1<=SLOTNUM<=6). SLOTNUM**==0** means that no free slot is available.

- If SLOTNUM **>0**, I move my car in front to the INDOOR, get out of the car and afterwards press a ***CARENTER*** button on the ***ParkServiceGUI***. Afterwards, the transport trolley takes over my car and moves it from the INDOOR to the selected parking-slot. The ParkServiceGUI will show to me a receipt that includes a (unique) ***TOKENID***, to be used in the *car pick up* phase.

As a **client - car pick up phase** :

- I intend to use the ParkServiceGUI to submit the request to pick up my car, by sending the TOKENID previously received.
- Afterwards, the transport trolley takes over my car and moves it from its parking-slot to the OUTDOOR-area.
- I move the car, so to free the OUTDOOR-area.

As a **parking-manager**:

- I intend to use the ***ParkServiceStatusGUI*** provided by the ParkManagerService to observe the ***current state*** of the parking area, including the value TA of the temperature, the state of the fan and the state of the transport trolley (**idle, working or stopped**).
- I intend to ***stop*** the transport trolley when **TA > TMAX**, activate the fan and wait until **TA < TMAX**. At this time, I stop the fan and resume the behavior of the transport trolley. Hopefully, the **start/stop of the fan** could also be automated by the ParkManagerService, while the **start/stop of the transport trolley** is always up to me.
- I expect that the ParkManagerService sends to me an ***alarm*** if it detectes that the OUTDOOR-area has not been cleaned within the DTFREE interval of time.

Requirements

The ParkManagerService should create the ParkServiceGUI (for the client) and the ParkServiceStatusGUI (for the manager) and then perform the following tasks:

- ***acceptIN***: accept the request of a client to park the car if there is at least one parking-slot available, select a free slot identified with a unique SLOTNUM.
  A request of this type can be elaborated only when the **INDOOR-area is free**, and the transport trolley is at home or working (**not stopped** by the manager). If the INDOOR-area is already engaged by a car, the request is not immediately processed (the client could simply wait or could - optionally - receive a proper notice).
- ***informIN***: inform the client about the value of the SLOTNUM.

If **SLOTNUM>0**:

1. ***moveToIn***: move the transport trolley from its current localtion to the INDOOR ;
2. ***receipt***: send to the client a receipt including the value of the TOKENID ;

3. ***moveToSlotIn***: move the transport trolley from the INDOOR to the selected parking-slot;
4. ***backToHome***: if no other request is present, move the transport trolley to its home location, else ***acceptIN*** or ***acceptOUT***.

If **SLOTNUM==0**:

- ***moveToHome***: if not already at home, move the transport trolley to its home location.

- ***acceptOUT***: accept the request of a client to get out the car with TOKENID. A request of this type can be elaborated only when the **OUTDOOR-area is free** and the transport trolley is at home or working (**not stopped** by the manager). If the OUTDOOR-area is still engaged by a car, the request is not immediately processed (the client could simply wait or could - optionally - receive a proper notice).
  1. ***findSlot***: deduce the number of the parking slot (***CARSLOTNUM***) from the TOKENID;
  2. ***moveToSlotOut***: move the transport trolley from its current localtion to the CARSLOTNUM/parking-slot ;
  3. ***moveToOut***: move the transport trolley to the OUTDOOR ;
  4. ***moveToHome***: if no other request is present move the transport trolley to its home location;
     else ***acceptIN*** or ***acceptOUT***

- ***monitor***: update the ParkServiceStatusGUI with the required information about the state of the system.

- ***manage***: accept the request of the manager to stop/resume the behavior of the transport trolley.

About the devices

All the sensors (weigthsensor, outsonar, thermometer) and the fan should be properly simulated by mock-objects or mock-actors.

When using a real robot

No further requirement.

When available a Raspberry and a sonar

The outsonar could be a real device. We can simulate the presence/absence of a car.

When using only the virtual robot or no real sonar available

Consider the new requirement:

- **authorize**: allow a manager to use the ParkServiceStatusGUI only if she/he owns **proper permissions**.

Requirement analysis

During a meeting with the client the following definitions are clarified:

- **ParkManagerService**: A computer system that meets the customer's requirements,it is composed of entities that communicate with each other and can potentially communicate with entities outside the system.
- **transport trolley**: An entity that can move around the parking area in the **permitted areas**, communicate with other information entities or receive commands, load and unload a car.
- **home**: Specific position of the trolley within the parking area. It is the starting position and the destination position in case no tasks are assigned to the trolley. In this position the trolley does not contain the car. Position (1,1) of map.
- **parking-area**: Permissible area for the trolley. Notable position : inaccessible positions called slots, positions beside the slots for loading and unloading cars, a position adjacent of INDOOR-area for loading the car from the outside of the system, an position adjacent OUTDOOR-area position for unloading towards the outside system, a home position.
- **INDOOR**: Abstract separation surface between the exterior of the system and the parking area. Separate the parking-area from the INDOOR-car position to perform the car loading(UPLOAD) operation.
- **OUTDOOR**: Abstract separation surface between the exterior of the system and the parking area. Separate the parking-area from the OUTDOOR-car position to perform the car unloading (DOWNLOAD) operation.
- **INDOOR-area**: Specific position outside the parking area delegated to the positioning of the car before the loading operation performed by the trolley from the contiguous position on this side of the INDOOR.
- **OUTDOOR-area**: Specific position outside the parking area delegated to the positioning of the car before the loading operation performed by the trolley from the contiguous position on this side of the OUTDOOR.This area must be cleared by the user within the time DTFREE.
- **weigthsensor**: Sensor that measures the weight of the car. It has the possibility to communicate the measurement performed.
- **weigth**: Car weight measurement.

- **outsonar**: Sensor that detects the presence / absence of the car. Has the ability to communicate presence / absence.
- **parking-slots**: Areas where it is possible to unload / load a car.The trolley cannot enter these areas.The number of these areas is limited to 6.The position of these areas (marked by inside the map) within the parking-area is defined by a file (parkingMap.txt)..
- **thermometer** : Sensor that measures the temperature of the car. It has the possibility to communicate the measurement performed..
- **TA**: Temperature value of thermometer.
- **fan**: Device delegated to decrease the temperature in the parking area. It must be controlled via the network by the system. It has two modes : on and off. It can be switched off automatically or following a command from the operator (parking-manager).
- **TMAX**: Thermometer temperature above which an alarm is triggered.
- **RD**: Elementary unit of measurement of the length of the map.For example, the space occupied by the trolley is an RD X RD, the space occupied by a slot is RD X RD.
- **fixed obstacles**: Impassable positions for the trolley on the map. In the parkingMap.txt file they are marked with an X at the border or at 1 RD from the border.
- **parking-manager**: Parking Supervisor. It has the right to: start \ stop of the trolley, start \ stop of the fan.Monitors: the temperature, the alarm for exceeding the exit parking time (DTFREE).
- **SLOTNUM**: Slot available where cars can be unloaded / loaded inside the parking area.The minimum value is 0, the maximum value is 6. SLOTNUM == 0 means no slot is avaliable.
- **CARENTER**: Name of the button to press after the customer has parked the car in the INDOOR-AREA.After pressing the button, the trolley, as soon as it is available,it will come to load the car and take it to the position in front of the established SLOT and then perform the unloading of the car..
- **ParkServiceGUI**: Graphic interface for the service user. It allows you to: request entry to the parking area (CARREQUEST button) , if there are slots available (SLOTNUM> 0) then a TOKENID containing a progressive number and the slot in which the car will be parked will be provided through a label, a button of CARENTER to start the parking process.
- **TOKENID**: String containing a unique progressive number and the slot number (SLOTNUM) where the car is parked. Identifies a car and the slot where it is parked. Regular Expression : "\d{3,}\-[1-6]" like 123-4 , it means progressive 123 at slot 4
- **ParkServiceStatusGUI** : Graphical interface for the manager. To see the functions of the GUI see the requirements of the parking-manager.
- **current state**: Set of variables that define the system such as: trolley position, weight sensor value, alarm for exceeding the TDFREE time in OUTDOOR-area, value of the TA temperature sensor, alarm for exceeding the TMAX temperature value, fan status (on \ off), Trolley status (idle, working, stopped), SLOTNUM value, occupancy status of each slot.
- **state of the fan**: Status of the fan. The possible states are: on, off.

- **start/stop of the fan**: Commands that can be sent to the fan. The commands that can be sent are: start, stop.
- **alarm** : Alarm that is displayed in the manager GUI when a car in the OUTDOOR-area does not free the position in DTFREE time. The alarm stops when the car clears the position.Note: the sensor at the OUTDOOR-park is responsible for detecting the presence / absence of the car.
- **CARSLOTNUM**: Number of slot where a car is parked.Duplicated entity with SLOTNUM?
- **acceptIN**: Following a customer's request (CARENTER): if a slot is free (SLOTNUM> 0) then the TOKENID is generated and inserted in the ParkQueue queue and the TOKENID is sent to the customer otherwise the TOKENID ("0") is generated and sent to the client.
- **start/stop of the transport trolley**: Commands that can be sent to the trolley. The commands that can be sent are: start, stop.

Additional concepts are added here :

- **CARREQUEST** :Button present in the ParkServiceGUI, which notifies the system the desire to enter the parking area.
- **FRONT-SLOT**: Position to the side of the slots from which it is possible to carry out the loading and unloading operations.
- **ParkQueue**: Queue containing the TOKENIDs of the cars : parked in the slots, currently in the INDOOR-Area waiting to be transported from the INDOOR-Area to the slots or currently on the trolley towards the slots, currently in the slots waiting to be transported from the slots to the OUTDOOR-Area or currently on the trolley in the direction of the OUTDOOR-Area or in the OUTDOOR-Area awaiting exit from the system.
- **permitted areas**: Area walkable by the trolley

Contradictions , Duplicated and overlap requirement :

- **SLOTNUM and CARSLOTNUM** : it seems describe the same concept
- **If SLOTNUM==0: moveToHome: if not already at home, move the transport trolley to its home location.**: in the case 5 slot are occupated and trolley it's performing moveToSlotIn task, trolley must go home replacing current task.Final state will be trolley at home with a loaded car. Any other task are impossible.

Device : after an interaction with the product-owner we get the information that all sensors are devices fixed to the wall and connected to the network through a raspberry. In the absence of other information it is assumed that they are all active devices communicating the change of state.

Sprint 1

Problem analysis

In the customer's requirements the map in the html file has one form and in the parkingMap.txt file it has another one, in the following analyzes the one in the file is taken as valid. The map has some ambiguities: for example the positions of the slots are not numbered, and neither are the positions of the loading / unloading areas. The team proposes to change the map symbols like this:

```
X,  X,  X,  X,  X,  X,  X,  X,
X,  H,  0,  0,  0,  0,  I,  X,
X,  0, A1, S1, S4, A3,  0,  X,
X,  0, A2, S2, S5, A5,  X,  X,
X,  0, A3, S3, S6, A6,  0,  X,
X,  0,  0,  0,  0,  0,  O,  X,
X,  X,  X,  X,  X,  X,  X,  X
```

Position :

- **H** : Home position of the trolley. Here trolley execute Sleep operation
- **I** : position from which can Upload a car
- **O** : position from which can Download a car
- **An** : front-slot **A**rea from which can Download\Upload a car
- **Sn** : **S** where cars are parked

The trolley stops only in fixed positions, continuous positions between two map coordinates are not possible. The trolley can perform the operation (OP) of walk (W), turn left (L), turn right (R), sleep (S), load (U) and unload (D). These characters (W, L, R, U, D, S) are part of the ROUTEHI alphabet, while the lower case version of the ROUTELO alphabet (see: Logic Architecture). In order to identify each position, through coordinates (x, y), of the map without ambiguity, new concepts are introduced:

All positions are divided into:

- **ReachablePoint**

The trolley can enter these positions. The accessible positions are those with the symbols H, 0, An, I and O. These positions are divided into:

- **KeyPoint**

  In these positions the trolley can be stopped to carry out the loading / unloading operations or Sleep in the case of H (Home).

  These positions are divided into:

  - **End-point**

    In these positions the trolley could receive a new Plan. In the positions of An it can carry out loading and unloading operations. (FRONT-SLOT) In the H position it can perform the sleep operation. In the O position it can perform the unloading operation.
    - **Avaliable-Point**

      In these positions the trolley receives a new Plan after the unloading or sleep operations or in other words is becames "Avaliable" after the D or S operation.
      In the An positions after the unloading operation, the trolley receives a new Plan .
      In the O position, after the unloading operation, the trolley receives a new Plan.
      In the position of H after the sleep operation it receives a new plan if it exists otherwise it remains in this position until a new Plan is received.
    - **No-End-Point**

      The only position is I and in this position the trolley cannot receive a new Plan.
      In the I position it can perform the loading operation.
  - **OnlyMovePoint**

    In these positions the trolley does not carry out operations of any kind, it can only transit.
- **UnReachablePoint**

  The inaccessible positions are those are marked with X and Sn.

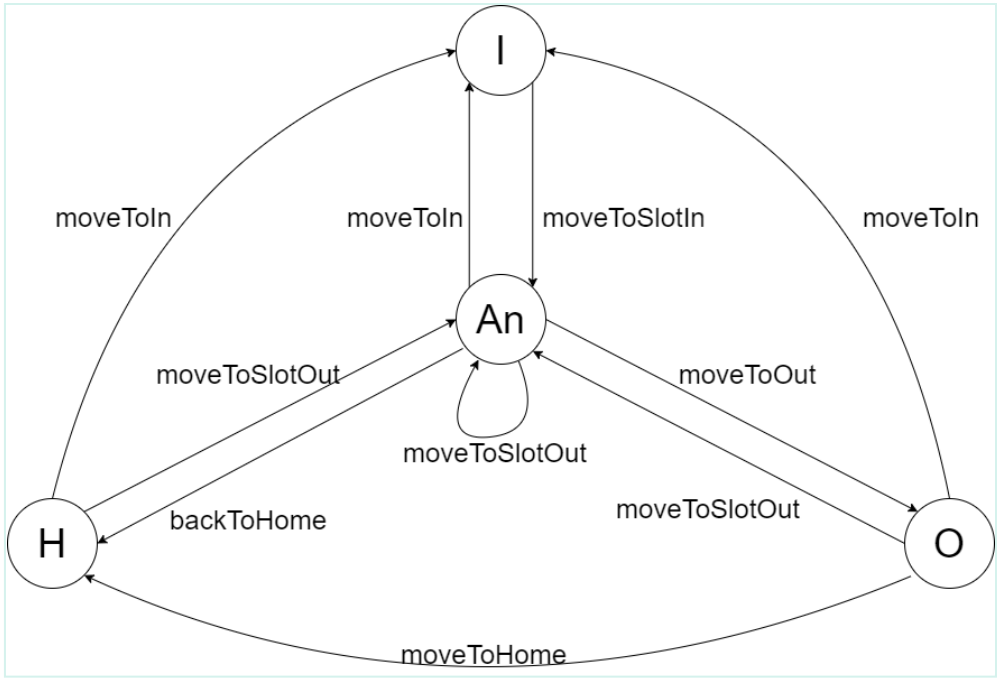
The system requires a research phase to determine the execution times of the operations like w. At the end of this phase, it may be necessary to introduce a trolley calibration every time it reaches the corner positions such as I, O and H. This calibration will exploit the object-detection functions of the virtual environment , due to this update logical architecture will be upgraded.

Table of SubPlan

			SubPlan							
		To								
		H	I	O	A1	A2	A3	A4	A5	A6
From	H		U (moveToIn)		U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)
	I				D (moveToSlotIn)	D (moveToSlotIn)	D (moveToSlotIn)	D (moveToSlotIn)	D (moveToSlotIn)	D (moveToSlotIn)
	O	N (moveToHome)	U (moveToIn)		U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)
	A1	N (backToHome)	U (moveToIn)	D (moveToOut)		U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)
	A2	N (backToHome)	U (moveToIn)	D (moveToOut)	U (moveToSlotOut)		U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)
	A3	N (backToHome)	U (moveToIn)	D (moveToOut)	U (moveToSlotOut)	U (moveToSlotOut)		U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)
	A4	N (backToHome)	U (moveToIn)	D (moveToOut)	U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)		U (moveToSlotOut)	U (moveToSlotOut)
	A5	N (backToHome)	U (moveToIn)	D (moveToOut)	U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)		U (moveToSlotOut)
	A6	N (backToHome)	U (moveToIn)	D (moveToOut)	U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)	U (moveToSlotOut)	

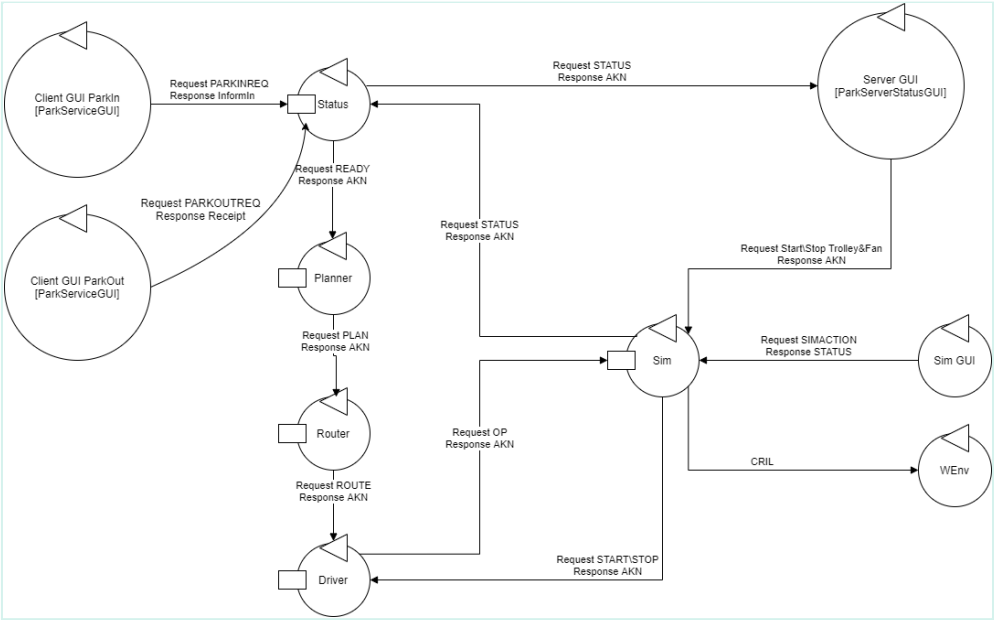
The SubPlan table defines the path of the trolley from a **KeyPoint** from to another **KeyPoint** to and the operation to be performed upon reaching the target KeyPoint and the name of the task. The consideration that can be made is that the tasks moveToIn, moveToSlotIn, backToHome, moveToHome, moveToSlotOut, moveToOut can be considered as Subplans. To complete a customer request, for example a pick up 2 Subplans are required. This sequence is called Plan, this implies that a Plan always starts from one **Avaliable-Point** and ends at another **Avaliable-Point**. The next table defines all possible Tasks.

Table of SubPlan



As already indicated in the requirements analysis (if SLOTNUM == 0 -> moveToHome), the existence of a requirement in contradiction with the condition that it is not possible to invoke a task that brings the trolley back to Home from any position, and whether or not it has the load, it leads at this point in the problem analysis to ignore this requirement.

Logic Architecture



The Status actor contains the state of the system. It implements the logic of the acceptIN, informIN, moveToIn, moveToSlotIn, backToHome, moveToHome, acceptOut, findSlot, moveToSlotOut, moveToOut and the mock-object monitor.
It is the only actor who maintains the internal state of the system, all the others are stateless.
Status sends a message to Planner the message READY [TOKENID, current position, destination].
Status contains a ParkInQueue for managing cars in the parking phase, a SlotList for parked cars, and a ParkOutQueue for cars in the collection phase.
Status write a log file for every state change : tasks ,message, sensor , operation , position , queue, list.

After receiving the READY, the Planner actor creates the PlanString (see: Plan table Plan column) to be sent to Route.

The Route actor sends the high-level route (ROUTEHI), a sequence of characters to control the trolley [W, R, L, U, D, S], to the Driver actor.

The Driver actor translates the ROUTEHI into the low-level route (ROUTELO) composed of a sequence of w, r, l, u, d, s followed by the execution time.
The Driver actor sends the pair r: routetime to Sim at a time with a routetime delay between one sending and the next. Sending can be suspended / resumed by a SIM message.

The Sim actor contains the mock-objects of: smartthermometer (thermometer + alarm), fan, weigthsensor, outsonar, manager.
Smartthermometer: mock-object that sends a message only when Ta> TMax or Ta <TMax to Status. Includes modeling of the thermometer and alarm management.
All mock-object send messages only on change status.
All mock-objects of the actor Sim send a message to Status.
The Sim actor commands WEnv with cril language(see : VR2021.pdf)

The Sim GUI is dedicated to modifying the system state (fan, thermometer, outsensor, weightsensor) and running tests.

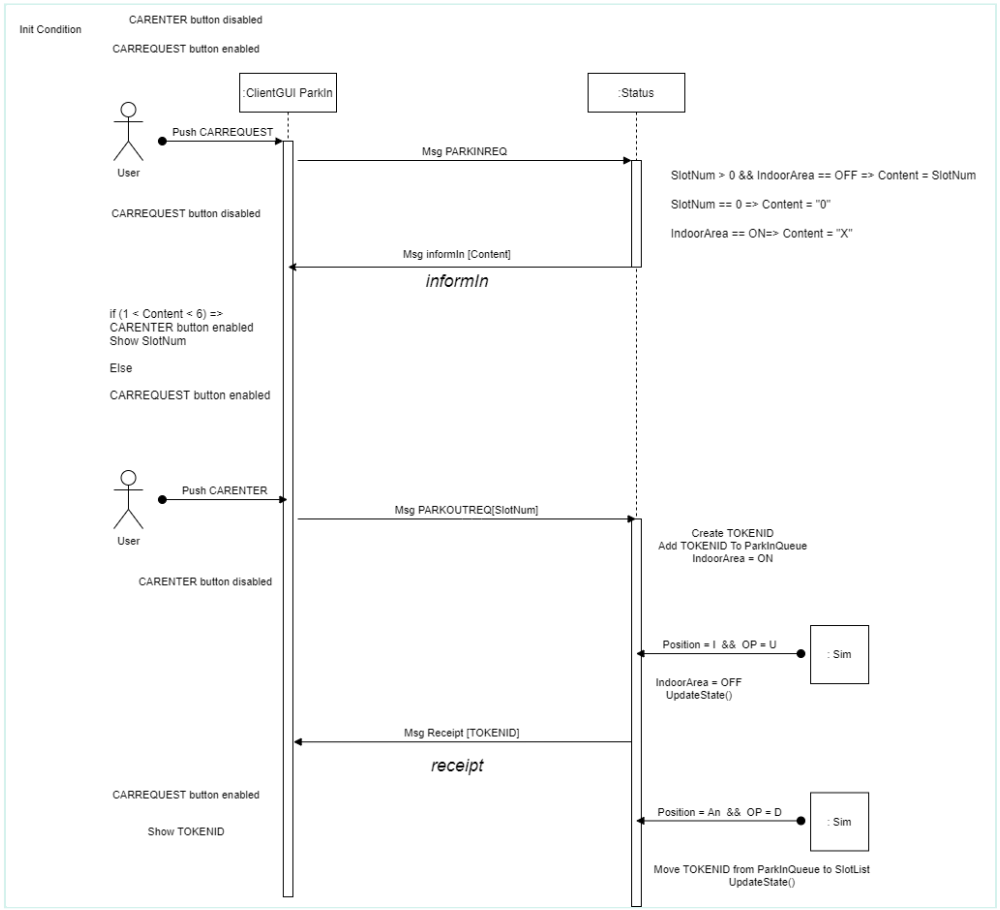
ServerGUI is enabled only after insert proper permission.

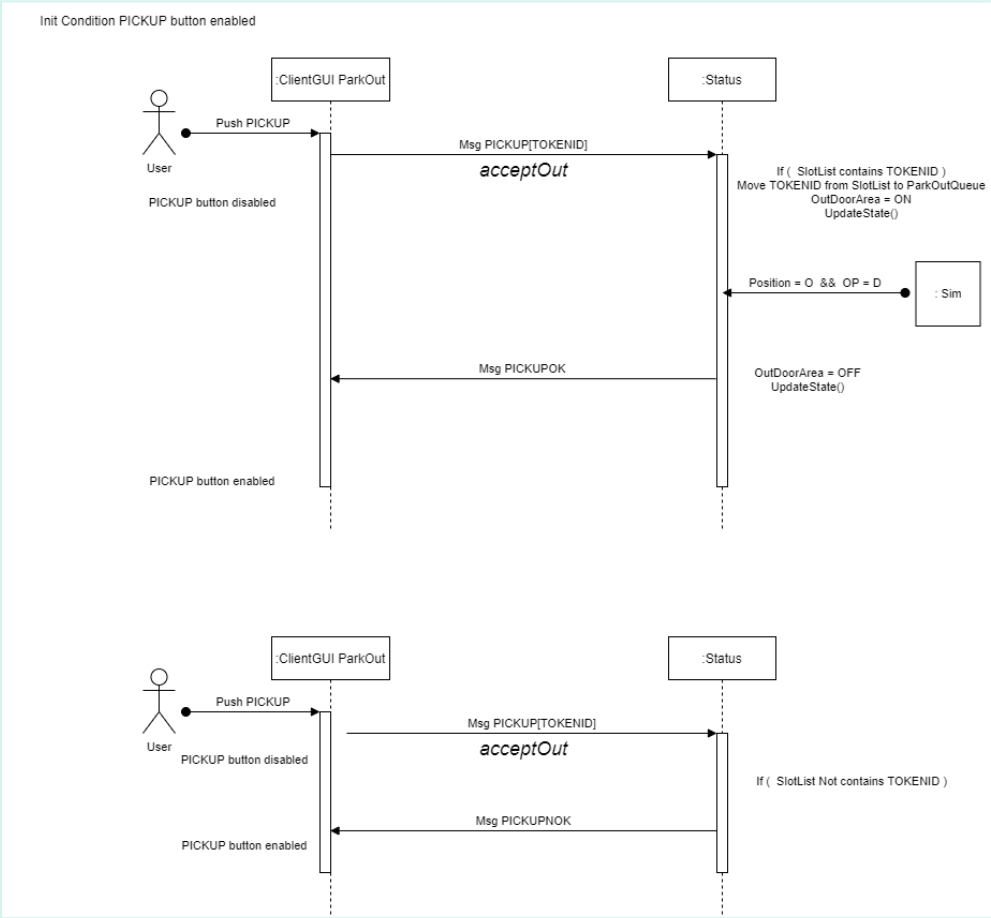
Table of Plans

N°	SlotFree >0	Position	Request	Plan	Test	Tasks					ParkInQueue	ParkOutQueue	OP	OutdoorArea
	F/T	I/O/H/N	I-M/M-O/H											
1	F	I	I-M	Null										
2	F	I	M-O	Null										
3	F	I	H	Null										
4	F	O	I-M	Refused										
5	F	O	M-O	O-M-U-O-D	5	acceptOUT	findSlot	moveToSlotOut	moveToOut		0	>0	D	Off
6	F	O	H	O-H	6	moveToHome					0	0	D	/
7	F	H	I-M	Refused										
8	F	H	M-O	H-M-U-O-D	8	acceptOUT	findSlot	moveToSlotOut	moveToOut		0	>0	S	Off
9	F	H	H	Null										
10	F	N	I-M	Refused										
11	F	N	M-O	N-M-U-O-D	11	acceptOUT	findSlot	moveToSlotOut	moveToOut		0	>0	D	Off
12	F	N	H	N-H	12	moveToHome					0	0	D	/
13	T	I	I-M	Null										
14	T	I	M-O	Null										
15	T	I	H	Null										
16	T	O	I-M	O-I-U-M-D	16	acceptIN	informIN	moveToIn	receipt	moveToSlotIn	>0	>=0	D	/
17	T	O	M-O	O-M-U-O-D	17	acceptOUT	findSlot	moveToSlotOut	moveToOut		0	>=0	D	Off
18	T	O	H	O-H	18	moveToHome					0	0	D	/
19	T	H	I-M	H-I-U-M-D	19	acceptIN	informIN	moveToIn	receipt	moveToSlotIn	>0	>=0	S	/
20	T	H	M-O	H-M-U-O-D	20	acceptOUT	findSlot	moveToSlotOut	moveToOut		0	>=0	S	Off
21	T	H	H	Null										
22	T	N	I-M	N-I-U-M-D	22	acceptIN	informIN	moveToIn	receipt	moveToSlotIn	>0	>=0	D	/
23	T	N	M-O	N-M-U-O-D	23	acceptOUT	findSlot	moveToSlotOut	moveToOut		0	>=0	D	Off
24	T	N	H	N-H	24	backToHome					0	0	D	/

The Plan table identifies under which conditions the Status actor sends the READY command to the Planner actor.
The 4 columns on the right , SlotFree and Position are the variables that identify the name of the Plan (PlanString) that Planner sends to the Router and the corresponding tasks that are performed.
This table has a 1: 1 correspondence with the TestsOfThePlans group of the Test Plan.
Tasks in blue are named MovingTask (see:Test Plan).

Sequence diagrams





Use Cases

Use Cases describe the sequence of messages between entities.
This list has a 1: 1 correspondence with the TestOfTheUseCases group of the Test Plan.

External

- 1.Car In
  - a)CARREQUEST : ClientGUI -> Status, Status=>ClientGUI
  - b)CARENTER : ClientGUI -> Status, Status=>ClientGUI, 8.a
  - c)FinalTestCARREQUEST : SimGUI ->Sim , Sim ->Status

- d)FinalTestCARENTER : SimGUI ->Sim , Sim ->Status , 8.a

- 2.Car Out
  - a)PickUp : ClientGUI -> Status, Status=>ClientGUI, 8.a
  - b)FinalTestPickUp : SimGUI ->Sim , Sim ->Status , 8.a
- 3.Fan
  - a)On Manual : ServerGUI ->Sim , Sim ->Status , Status-> ServerGUI
  - b)On Simulated : SimGUI-Sim, Sim-> Status, Sim->Driver ,Status-> ServerGUI
  - c)Off Manual : ServerGUI ->Sim , Sim ->Status , Sim->Driver , Status-> ServerGUI
  - d)Off Simulated : SimGUI-Sim, Sim-> Status, Sim->Driver , Status-> ServerGUI
- 4.Temp (Alarm)
  - a)On Simulated : SimGUI-Sim, Sim-> Status, Sim->Driver , Status-> ServerGUI
  - b)Off Simulated : SimGUI-Sim, Sim-> Status, Sim->Driver , Status-> ServerGUI
- 5.Weigthsensor
  - a)On Auto : ClientGUI -> Status, Status-> ServerGUI
  - b)Off Auto : Sim -> Status, Status-> ServerGUI
- 6.Outsonar
  - a)On Auto : Sim -> Status, Status-> ServerGUI
  - b)Off Auto : Sim -> Status, Status-> ServerGUI
  - c)OverTime Simulated : SimGUI=>Sim, Sim -> Status, Status-> ServerGUI
- 7.Trolley
  - a)On Manual : ServerGUI ->Sim , Sim ->Status , Sim->Driver , Status-> ServerGUI
  - b)Off Manual : ServerGUI ->Sim , Sim ->Status , Sim->Driver , Status-> ServerGUI
- Internal
  - 8. Execute Plan
    - a)Auto : Status -> Planner , Planner -> Driver , Driver ->Sim , Sim ->Status , Sim->WEnv , Status->ServerGUI
    - b)Manual [Simulated]:SimGUI-Sim, Sim -> Status , 8.a

Test Plan

The test plan is divided into 2 groups: the tests concerning the execution of the 13 Plans (see: Plan Table) and the tests concerning the 21 use cases.
Starting and ending position are the following:
O position means that trolley is in O position heading bottom after a D operation or empty.
H position means that trolley is in H position heading bottom after a S operation.
I position means that trolley is in I position heading top after a U operation.
An position with 1<n<3 means that trolley is in An position heading right after a D operation or empty.
An position with 4<n<6 means that trolley is in An position heading left after a D operation or empty.

Test form Plan Table

N°	Name	Initial Condition	Goal	Description	Succes
1	Test N = 5 Table of Plans	n=1,Trolley in O, no slot free	Execute MovingTasks (To An then O)	Push Test 1 in SimGUI	Trolley in O, slot An free others occupated
2	Test N = 6 Table of Plans	Trolley in O, no slot free	Execute MovingTasks (To H)	Push Test 2 in SimGUI	Trolley in H, no slot free
3	Test N = 8 Table of Plans	n=2,Trolley in H, no slot free	Execute MovingTasks (To An then O)	Push Test 3 in SimGUI	Trolley in O,slot An free others occupated
4	Test N = 11 Table of Plans	n=3,Trolley in An, no slot free	Execute MovingTasks (To An then O)	Push Test 4 in SimGUI	Trolley in O,slot An free others occupated
5	Test N = 12 Table of Plans	n=4,Trolley in An, no slot free	Execute MovingTasks (To H)	Push Test 5 in SimGUI	Trolley in H, no slot free
6	Test N = 16 Table of Plans	n=5,Trolley in 0, A1 is free	Execute MovingTasks (To I then An)	Push Test 6 in SimGUI	Trolley in An
7	Test N = 17 Table of Plans	n=6, Trolley in 0, An is occupated	Execute MovingTasks (To An then O)	Push Test 7 in SimGUI	Trolley in 0
8	Test N = 18 Table of Plans	n=1, Trolley in 0, An is occupated	Execute MovingTasks (To H)	Push Test 8 in SimGUI	Trolley in H, slot unchanged

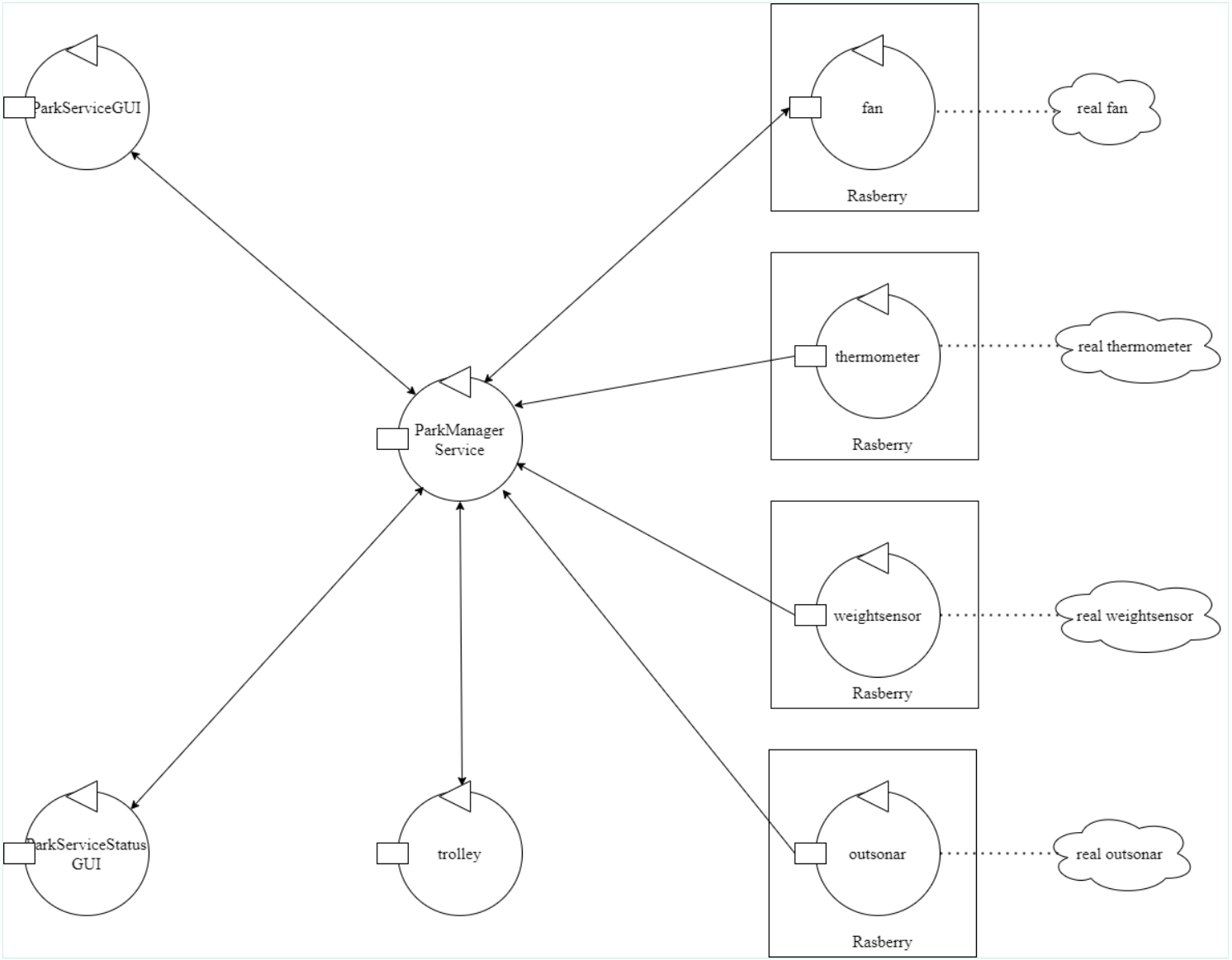
9	Test N = 19 Table of Plans	n=2, Trolley in H, An is free	Execute MovingTasks (To I then An)	Push Test 9 in SimGUI	Trolley in An, An is occupated
10	Test N = 20 Table of Plans	n=3, Trolley in H, An is occupated	Execute MovingTasks (To An then O)	Push Test 10 in SimGUI	Trolley in O, An is free
11	Test N = 22 Table of Plans	n=4,m=5, Trolley in An, An is occupated,Am is free,	Execute MovingTasks (To An then I then Am)	Push Test 11 in SimGUI	Trolley in Am,An is occupated, Am is occupated
12	Test N = 23 Table of Plans	n=6,m=1, Trolley in An, An is occupated,Am is occupated,	Execute MovingTasks (To An then Am then O)	Push Test 12 in SimGUI	Trolley in O, An is occupated, Am is free
13	Test N = 24 Table of Plans	n=2, Trolley in An, An is occupated	Execute MovingTasks (To H)	Push Test 13 in SimGUI	Trolley in H, An is occupated

Test form Use Case

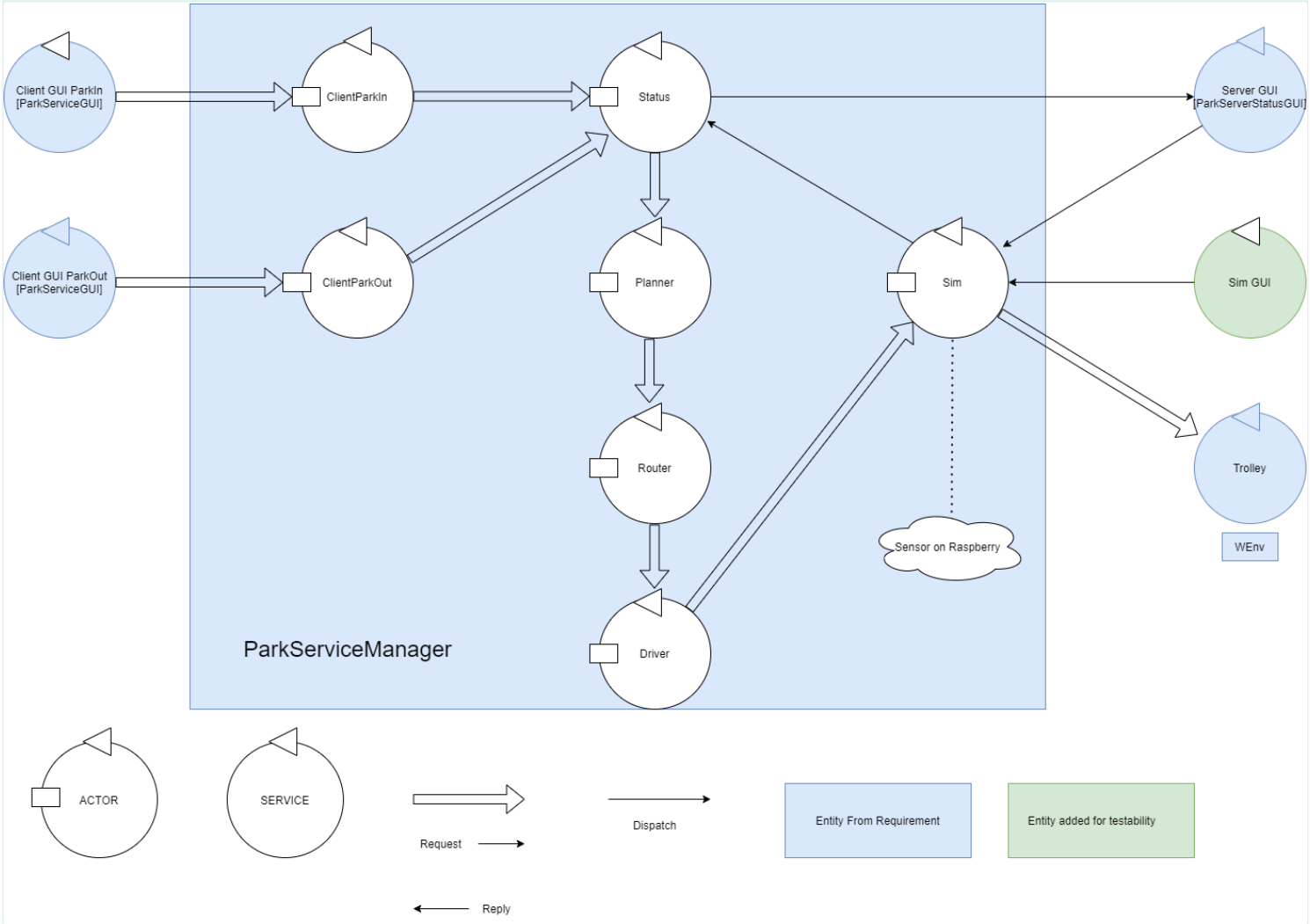
N°	Name	Initial Condition	Goal	Description	Succes
1	1.Car In.a.CARREQUEST	weightsensor=off,some slot free	Parkin phase CARREQUEST	Push CARREQUEST in ClientGUI	ClientGUI show slotnum
2	1.Car In.a.CARENTER	After a CARREQUEST	Parkin phase CARENTER	Push CARENTER in ClientGUI	ClientGUI show TOKENID
3	1.Car In.a.FinalTestCARREQUEST		See test "AL TERMINE DEL LAVORO"	Push FinalTestCARREQUEST in SimGUI	See Status log for state change
4	1.Car In.a.FinalTestCARENTER		See test "AL TERMINE DEL LAVORO"	Push FinalTestCARENTER in SimGUI	See Status log for state change
5	1.Car In.a.FinalTestPickUp		See test "AL TERMINE DEL LAVORO"	Push FinalTestPickUp in SimGUI	See Status log for state change
6	1.Car In.a.FinalTestComplete		See test "AL TERMINE DEL LAVORO"	Push FinalTestComplete in SimGUI	See Status log for state change

Sprint 1

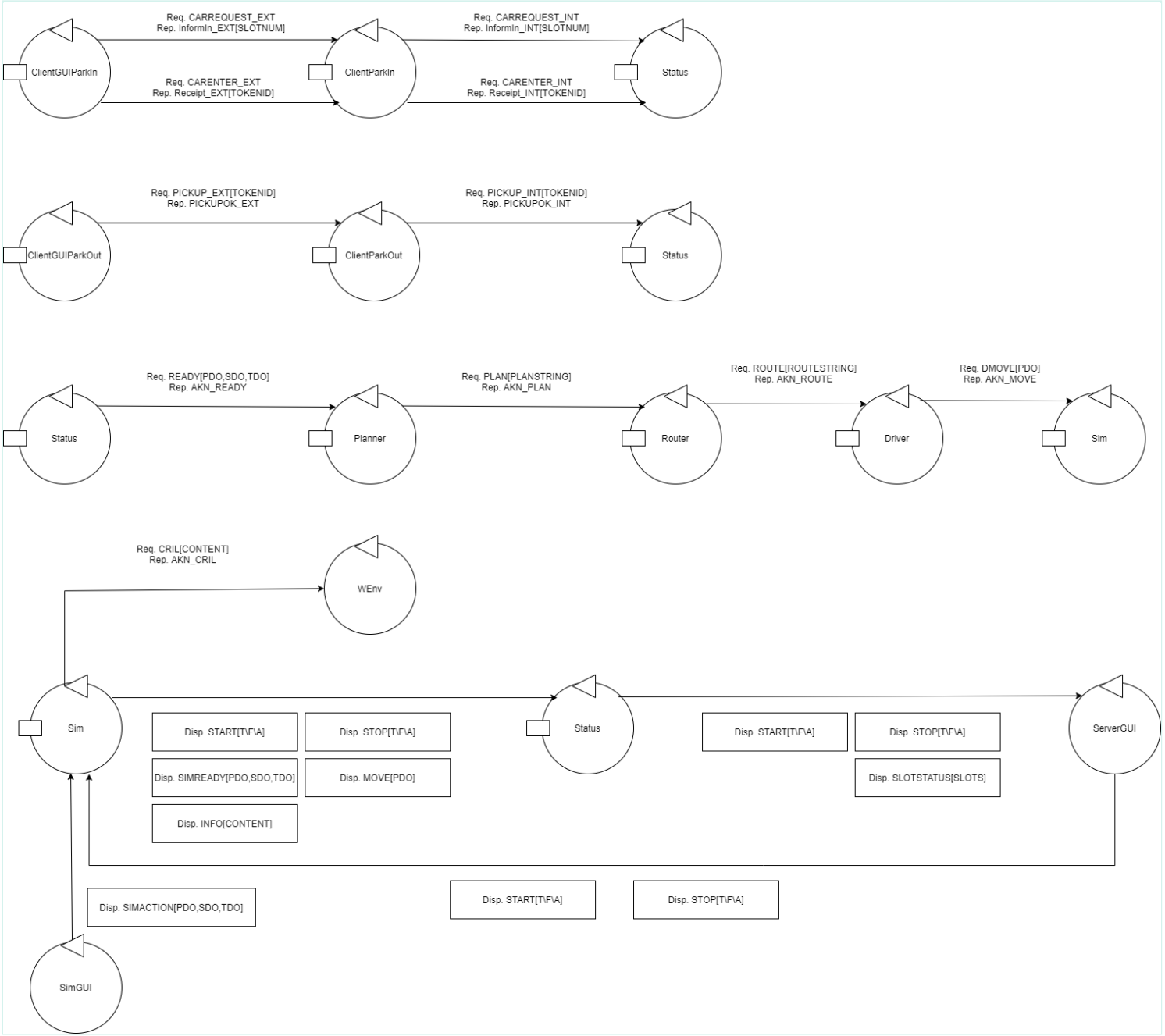
Model Of Requirements



Logic Architecture Main



Logic Architecture Details



In Logic Architecture Details actor are Qakactor.

Model

First ,and incomplete, version of model is defined in a qak file (see : parkingManagerServiceModel.qak)

Test Plan

The Test Plan contains 2 tests for the execution of the main functions of the application.
The ParkIn phase and the ParkOut phase.
In both cases the trolley starts from the Home position and goes to the Home position.
The tests are carried out by some actors, respectively clientparkintest and clientparkouttest (see : parkingManagerServiceModel.qak).

ParkIn Phase
Initial conditions: weightsensor = off, all slot free.
The tasks of acceptIN, informIN, moveToIn, receipt, moveToSlotIn, backToHome are performed.
Success conditions at the end of the test: the TOKENID is generated, slot 1 is occupied, the trolley is in the HOME position.

ParkOut Phase
Initial conditions: weightsensor = off, slot 1 occupied.
The acceptOUT, findSlot, moveToSlotOut, moveToOut, moveToHome task is executed.
Success conditions at the end of the test: all slots are free, the sonar has detected a presence and before DTFREE detects the absence of the car, the trolley is in the HOME position.

mail : ugo.marchesini@studio.unibo.it