LABORATORIO DI INGEGNERIA DEI SISTEMI SOFTWARE

Introduction

Requirements

[TFBO21ParkingISS.pdf](#)

Requirement analysis

Meaning of nouns and verbs

NOUNS:

- **DDR Robot**: Differential Drive Robot. It is capable of moving in the environment by executing simple move commands. It can be equipped with a range of devices. It must be capable of moving object as a transpot trolley with a squared form of fixed side length;

- **Home location**: fixed starting position inside the parking-area where the robot will be located before the software starts and where it will return if no jobs are pending;

- **Parking-area**: empty room that must include an indoor, indoor-area, outdoor, outdoor-area, parking-slots, thermometer and a fan;

- **Indoor**: used to get the car into the parking-area;

- **Indoor-area**: faces the indoor and hosts the car that needs to be parked. It is also equipped with a weightsensor;

- **Weightsensor**: sensor that measures the weigth of the car inside the indoor-area;

- **Outdoor**: used to exit the car from the parking-area;

- **Outdoor-area**: faces the outdoor and hosts the car to be picked up. It is also equipped with a ousonar. Once it is engaged by a car it should be freed within a prefixed interval of time;

- **Outsonar**: sonar that detects the presence of a car within a certain space;

- **Parking-slot**: area used for parking cars. The area covered by the parking-slots connot be crossed by the robot;

- **Thermometer**: tool used to measure the temperature inside the parking-area;

- **Fan**: tool used to lower the temperature inside the parking-area.

- **Map**: it is a representation of the parking-area. It is a grid of squares of fixed side length. It includes the parking-slot positions (marked with a red X) and the fixed obstacles in the area (marked with a black X);

- **Fixed obstacle**: non-moving object inside the parking-area that the robot cannot cross;

- **Wall**: fixed obstacle that delimits the parking-area;

ParkingServiceGUI: a user interacting component that allow the client to manifest the interst of entering the car into the parking-area and receiving information abount the state of the parking-area (ei. free slots). Also allow the client to call the robot to park the car;

- **ParkServiceStatusGUI**: a user interacting component that allow the parking-manager to observe the status of the parking-area including temperature and status of the fan and the robot;

- **Status**: information about the state of the robot, the fan, the temperature of the room, the car waiting for entering/exiting the parking-area, free/occupied parking-slots;

- **Token**: a unique identifier of the couple car-slot inside the parking-area;

- **Client**: fisical person that posses a car;

- **TMAX**: fixed max temperature above which the fan must be activated;

VERBS:

- **Stop the robot**: stops the robot in the position it is currently. This operation is always up to the parking-manager;

- **Start the robot**: start the robot from the position it is currently. This operation is always up to the parking-manager;

- **Stop the fan**: stops the fan from working. This operation can either be automated (by checking the temperature) or not;

- **Start the fan**: start the fan. This operation can either be automated (by checking the temperature) or not;

- **Pick up car**: the client who parked the car must be able of collecting it from the outdoor-areaby entering the token related to his/her car;

- **Enter car**: the client must be able to enter the car inside the indoor-area once it is free amd he/she must receive as answare the number of free parking-slots and if a slot is available he/she will receive a unique token;

- **Check status**: the manager must be able to check the status of the entire system;

- **Inform the client**: send/show a massage to the client;

- **The start/stop of the fan could also be automated**: after a further analysis it has been decided that the start and stop of the fan are both automated and up to the manager;

Definition of a formal model of the requirements

Since our software must move the robot inside the parking-area, we must find a way to automatically test all tasks concerning movements.

At this stage, we need to introduce a **proper representation** of the parking-area. Luckly, the customer already gave us a file representing the parking-area as a *map*: parkingMap.txt
As we can see from the requirements, the map is a **grid of square with fixed size**:

							X
r							X
		X	X				X
		X	X				X
		X	X				X
							X
X	X	X	X	X	X	X	X
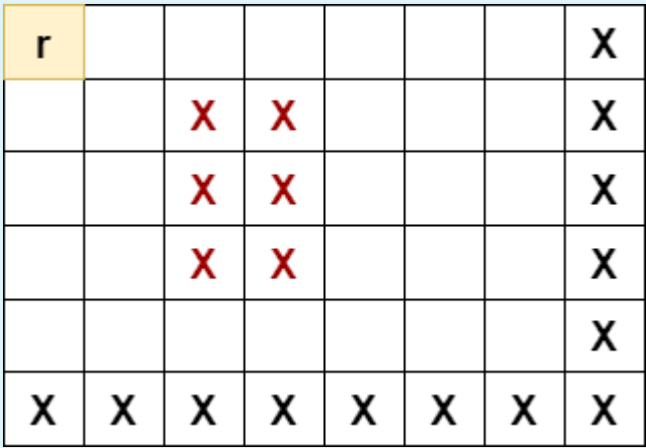
Scheme of the map as a grid

```
|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
```

Map given by the customer

In the above representation, the customer specify that:

- **r** : cell occupied by the robot
- **0** : cell not occupied (robot can pass over it)
- **X** : cell representing a paking-slot
- **X** : cell occupied by fixed obstacle
- ▢ : home

Overview of the requirements

The customer specifies all the devices and the *reserved-areas* of the parking room. Now, we introduce a first overview of all entities required by the customer:
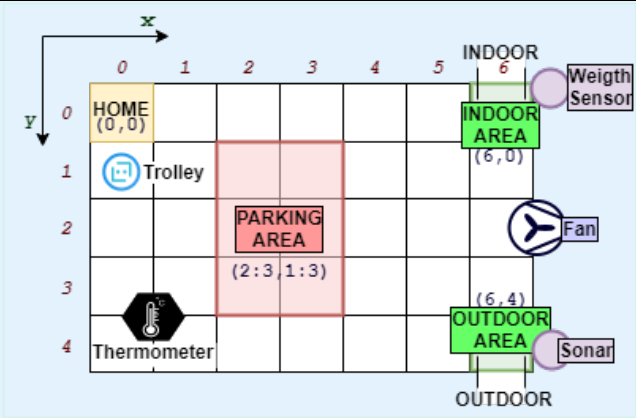
Graphical overview of the requirements
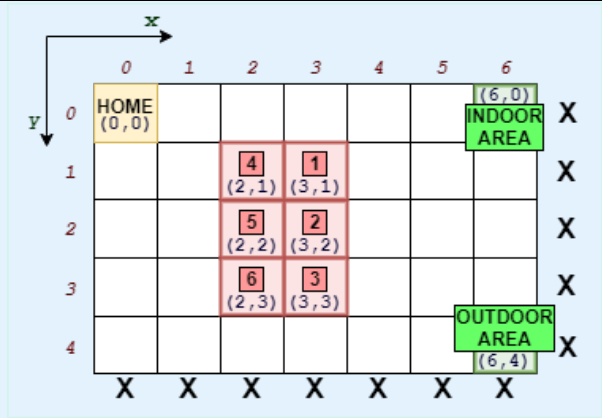
Reserved areas of the parking-room

We have introduced a *cartesian plan* over the room in order to define a *reference system*. Then, as indicated in the image above, the reserved areas are:

- **HOME** in the cell (0, 0)
- **Indoor-area** in the cell (6, 0)
- **Outdoor-area** in the cell (6, 4)
- **Parking-slots** in the cells (2:3, 1:3)

Customer also specify that the robot can walk over all cell of the parking-room except over the parking-slots. Zooming into the parking-area, we decide to numerate the available parking-sloot as indicated in the image above. Instead, about the entities:

- **Trolley**: customer specifies that it is a *DDR Robot* and also give us the needed software to build it at **it.unibo.qak21.basicrobot**;

- **Thermometer**: the customer does not provide any software, so we must manage it;

- **Weight Sensor**: the customer does not provide any software, so we must manage it;

- **Fan**: the customer does not provide any software, so we must manage it; however, he give us the code to turn on/off a led connected to a Raspberry (it.unibo.rasp2021/resources/Raspberry/bls) that can be adapted to work with a fan. Ideed, as a led, the unique operations that the software must provides are **turnOn( )** and **turnOff( )** in order to power on or power off the phisical fan.

- **Sonar**: the customer provides us the code SonarAlone.c: this C program continuously reads the distance from a phisical sonar connected to a Raspberry and print it to the standard output. In order to make this code more useful, we have adapted it in a previous developement:

SonarAlone.c (project *RaspiTools*) For a demo, see CSonar.java	1. allows the argument **-t** to specify the TRIG pin of the phisical device; 2. allows the argument **-e** to specify the ECHO pin of the phisical device; 3. allows the argument **-d** to specify the time must padd between one measurement and another; 4. allows the argument **-s** to specify a single measurement; 5. allows the argument **-i** to specify the possibility to control the cicle of measurement with an UNIX signal (*SIGUSR1*). For example, if we have a Sonar with ECHO and TRIG pin respectively connected to the pins 0 and 2 of a Raspberry, the invocation to get a single measurement is: <code>./SonarAlone -e 0 -t 2 -s</code> This invocation prints the value readed by the phisical sonar to the standard output.

mapUtil

A proper support for the room-map can be given by the following utility:

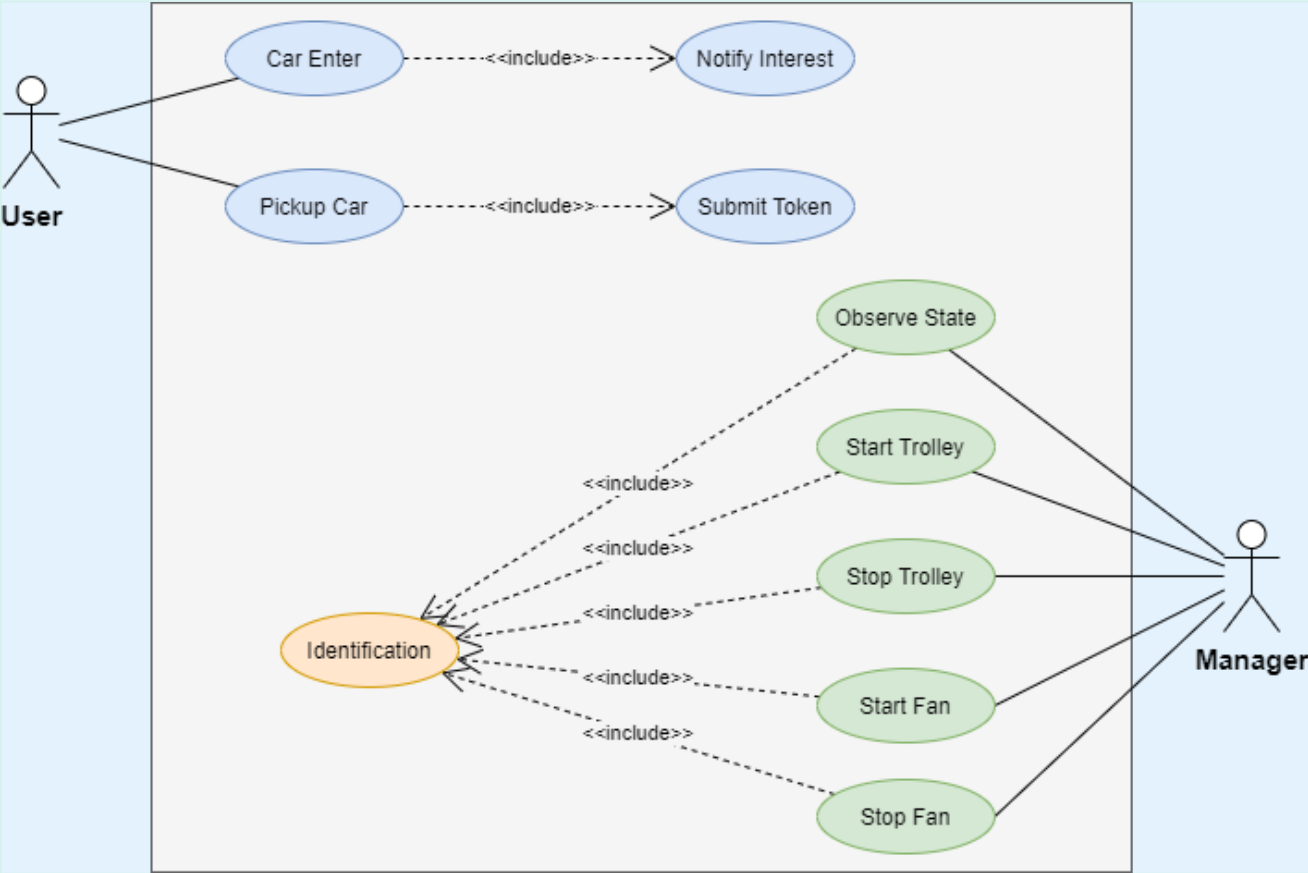|	
mapUtil.kt	1. creates a room map as a singleton; 2. provides the operation **doMove(move: String)** that updates the map according to the given robot move ($w\|s\|r\|l\|h$) 3. provides the operation **getMapAndClean( ):**

(project *it.unibo.kotlinSupports*) For a demo, see MainMapUsage.kt	**String** that returns a String-representation of the current map and resets the map to the empty map; 4. provides the operation **showMap( )** that prints on the standard output the String-representation of the current map.

This utility can be adapted to the map given by the customer.

Use Case

From the user stories provided by the customer it is possible to produce an Use Case Diagram:

Zooming into this diagram, we present some sample details about all use cases:

Actor: User	
	Car Enter • **Dependencies:** *-Notify Interest* • **Pre Condition:** *-The user recieve a slotnum higher*

than 0;
- **Post Condition**
  -The car is parked;
  -The transport trolley can go back to its home position or fulfill another request;
  -The user has received the tokenId
- **Main Scenario:**
  1. The user sends the slotnum previously received to the service;
  2. The user waits for a notification from the system that the indoor-area is free within a certain period of time;
  3. The user drive the car into the indoor-area and exits the car;
  4. The user press the CARENTER button;
  5. The system checks the positive presence of the car in the indoor-area;
  6. The client receive a unique token (use case Emit TokenId) to pick up the car from the outdoor-area;
  7. The transport trolley moves the car from the indoor-area to the selected parking-slot;

- **Alternative Scenario:**
  Scenario a: The user does not move the car into the indoor-area within a fixed period of time.
  2. The user loses the right to park the car and receive a proper notification;
  3. The system releases the reserved parking-slot;

  Scenario b: The user enters the indoor-area but does not press the CARENTER button within a fixed period of time (ITOCC).
  3. The client receive a unique token (use case Emit TokenId) to pick up the car from the outdoor-area;
  4. The transport trolley automatically parks the car;

Notify Interest

- **Dependencies:**
- **Pre Condition:**
- **Post Condition:**
  -The user receive the slotnum;
- **Main Scenario:**
  1. The user notify to the system the interest of entering his/her car inside the parking-area and adds to the request some useful information;
  2. The user receive a massage containing a slotnumber;
  3. The slot number received is higher than zero and representing the parking-slot number;

- **Alternative Scenario**:
  3. The slot number is equal to zero meaning the parking-area is full. In that moment, the car cannot enter.

Scenario c: The user does not enter the indoor-area but press the CARENTER button.
2. The system checks that the indoor-area is empty;
3. The system sends a proper notification to the user saying the car must be inside the indoor-area;

Scenario d: The indoor-area is already engaged.
2. The system checks that the indoor-area is already engaged then asks the client for waiting;
3. When the indoor-area is free, the system sends a notification to the client for entering the car;
4. Continue from point 3 of the main scenario

Submit Token

- **Dependencies:**
- **Pre Condition**
- **Post Condition**
  -The car is dropped in the outdoor-area;
  -The transport trolley can go back to its home position or fulfill another request;
- **Main Scenario**:
  1. The system checks the existance of the token entered and gets the relative parking-slot
  2. The system checks the outdoor area is free;
  3. The system notifies the user to wait the car at the outdoor-area;
  4. The transport trolley takes over the car and moves it from the parking-slot to the outdoor-area;
  5. The system notifies the user to pick up the car from the outdoor-area.

Pickup Car

- **Dependencies:**
  -Submit Token
- **Pre Condition**
- **Post Condition**
  -The outdoor-area is free;
- **Main Scenario**:
  1. The client can pick up the car from the outdoor-area within a fixed period of time (DTFREE);
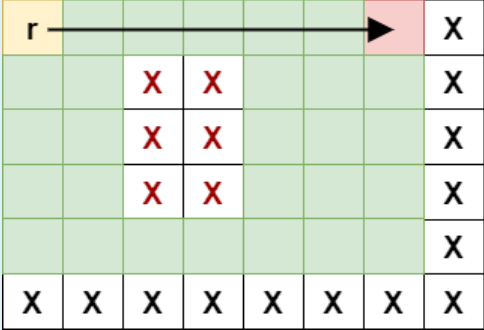- **Alternative Scenario:**

- **Alternative Scenario:**
  Scenario a: The token does not exist in the system.
  2. The system asks for a valid token;

  Scenario b: The outdoor-area is occupied by another car.
  2. The system notifies the user to wait;
  3. When the outdoor-area is free, the system notifies the user to wait the car at the outdoor-area;
  4. The transport trolley takes over the car and moves it from the parking-slot to the outdoor-area;
  5. The system notifies the user to pick up the car from the outdoor-area.

Scenario a: The fixed period of time to pick up the car runs out.
1. A notification is sent to the manager;
2. The manager free the outdoor-area and a proper notification should be sent to the user;

Actor: Manager

Identification

- **Dependencies:**
- **Pre Condition**
- **Post Condition**
- **Main scenario**:
  1. The manager enters the proper credential to access the system;
  2. The system after checking the credential opens the proper home page

- **Alternative Scenario:**
  Scenario a: Credential not recognized.
  2. The system shows the identification screen again;

Start Fan

- **Dependencies:**
  -Identification
- **Pre Condition:**
  -The fan is stopped;
- **Post Condition**: -The fan is started;
- **Main Scenario**:
  1. The fan starts and decrease the temperature inside the parking-area. This use case could be automated;

- **Alternative Scenario:**

Stop Fan

Start Trolley

- **Dependencies:**

Dependencies:
-*Identification*
- **Pre Condition:**
  -*The fan is working;*
  -*The temperature inside the parking-area is below TMAX;*
- **Post Condition**: -The fan is stopped;
- **Main Scenario**:
  1. The manager stops the fan;

- **Alternative Scenario:**

-*Identification*
- **Pre Condition:**
  -*The transport trolley is stopped;*
- **Post Condition**
  : -*The transport trolley is resumed;*
- **Main Scenario**:
  1. The transport trolley is resumed by the manager and continue to serve the current request, starts a new one or go back to the home position;

- **Alternative Scenario:**

Stop Trolley

- **Dependencies:**
  Identification
- **Pre Condition:**
  -*The transport trolley is working;*
- **Post Condition**
  : -*The transport trolley is stopped;*
- **Main Scenario**:
  1. The transport trolley is stopped from the manager;

- **Alternative Scenario:**

Observe State

- **Dependencies:**
  Identification
- **Pre Condition:**
- **Post Condition:**
  -*The status is shown*
- **Main Scenario**:
  1. The system's status is shown to the manager;

- **Alternative Scenario:**

About the fan, we specify that the fan must be automatically activated by the system if the temperature is abouve TMAX and speculary stopped when the temperature is below TMAX. Nonetheless, as specified in the use cases, the manager can manually start the fan if not already, and stop it only if the temperature is below TMAX.
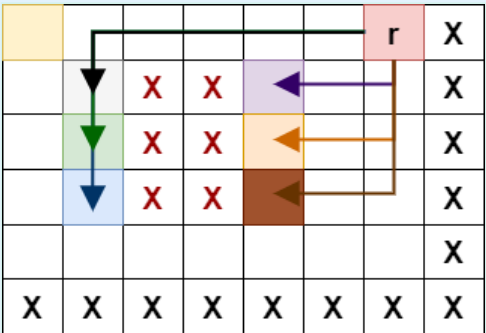
Task Modeling

moveToIn Task

- **Relative Use Case**: *Car Enter*
- **Legend:**

  o   : location where you can find the robot at the task call;

  o   : indoor-area.

- **Description**: This task is used by the system for

moving the trasport trolley towards the indoor area in order to pick up the car and park it.
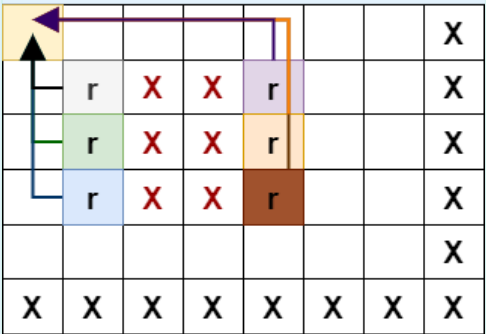
moveToSlotIn Task

- **Relative Use Case**: *Car Enter*
- **Legend:**

  - ▢ : location where you can find the robot at the task call;
  - Colored Square: cell used by the transport trolley to park the car in the adiacent parking-slot.

- **Description**: This task is used by the system for parking a car to a specified parking-slot.
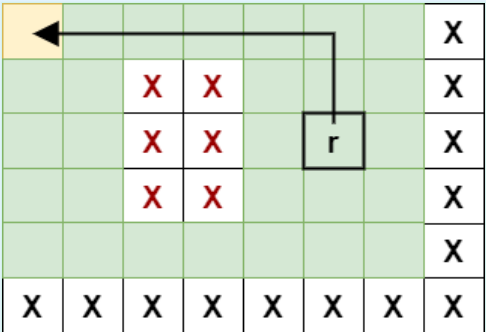
backToHome Task

- **Relative Use Case**: *Car Enter*
- **Legend:**

  - Colored Square: location where you can find the robot at the task call.

- **Description**: This task is used by the system for returning the transport trolley to the home after it has parked a car and there are not pending requests.

moveToHome Task

- **Relative Use Case**: *Submit Token*
- **Legend:**

  - ▢ : location where you can find the robot at the task call.

- **Description**: This task is used by the system for moving the trasport trolley to the home.
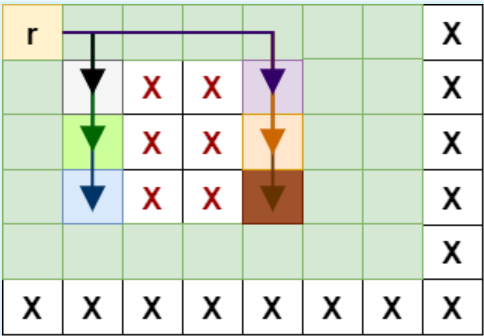
moveToSlotOut Task

- **Relative Use Case**: *Submit Token*

- **Legend:**

  - ☐ : location where you can find the robot at the task call;
  - Colored Square: cell used by the transport trolley to pick up the car from the adiacent parking-slot.

- **Description**: This task is used by the system for picking up a car in order to transport it to the outdoor-area.
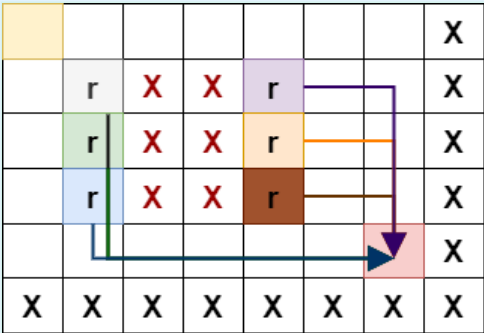

---

moveToOut Task

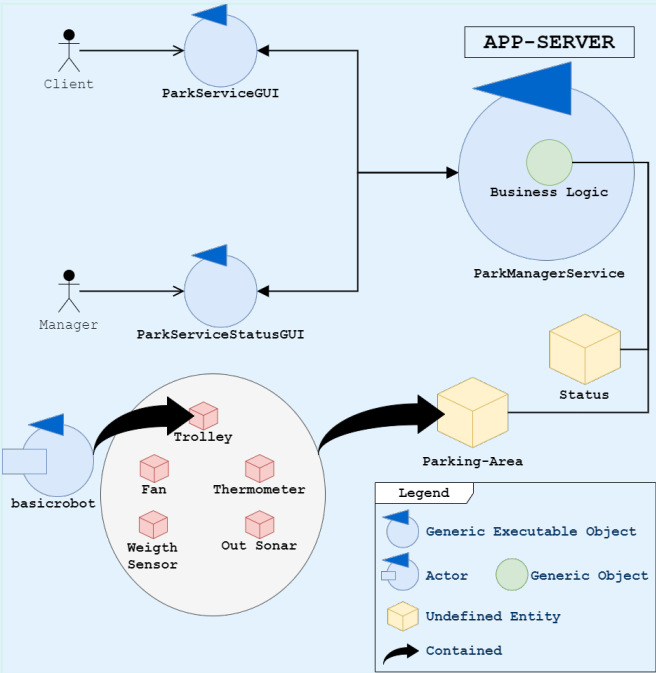- **Relative Use Case**: *Car Enter*
- **Legend:**

  - ☐ : outdoor-area;
  - Colored Square: location where you can find the robot at the task call.

- **Description**: This task is used by the system for transporting a car to the outdoor-area.
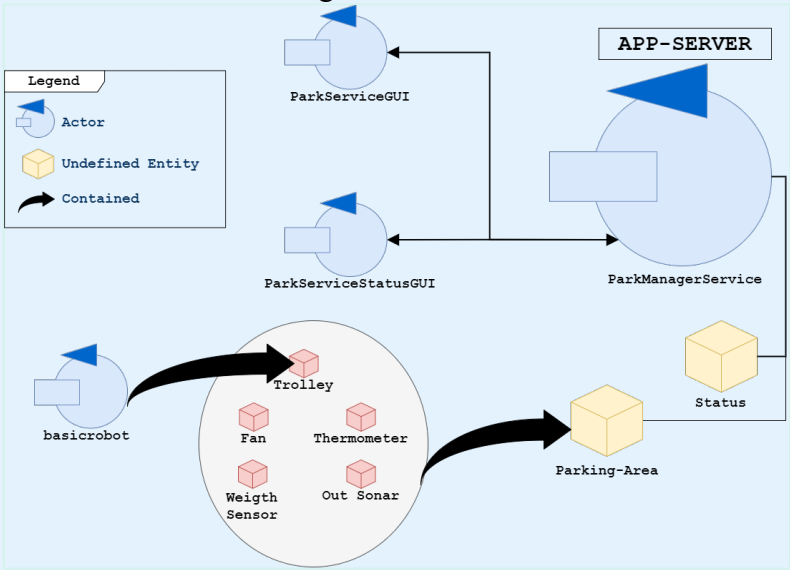

Problem analysis
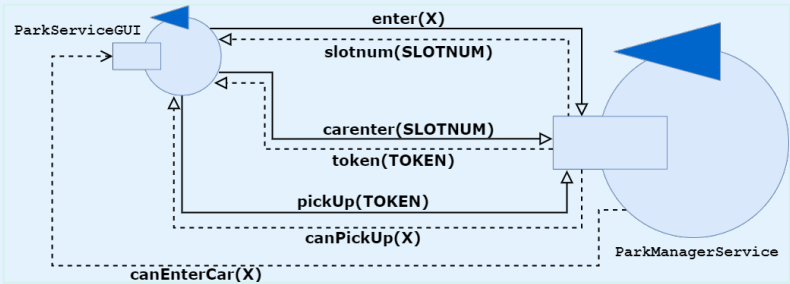
First Logical Architecture

QAK - Metamodeling

It is useful to adopt the **QActor (meta)model** to provide an executable model for the logical architecture. By adopting this meta-model we can map the General Executable Object by as QAK-Actors in order to produce a more detailed logical architecture:

Interaction ParkManagerService-ParkServiceGUI

Message Type Analysis

enter(X)	slotnum(SLOTNUM)

- **Type:** Request;
- **From:** ParkServiceGUI;
- **To:** ParkManagerService;
- **Reply Message:** slotnum
- **Payload:**
  - **X:** generic informational message;
 	
- **Explanation:** From the requirements it is known the need from the user to notify his/her interest to enter the parking-area;
	- **Type:** Reply;
	- **From:** ParkManagerService;
	- **To:** ParkServiceGUI;
	- **Request Message:** enter
	- **Payload:**
	  - **SLOTNUM:** number between 0 and "number of slots";
	
	- **Explanation:** From the requirements it is known that after a user notify his/her interest to enter the parking-area, the service sends him/her the slotnumber;

carenter(SLOTNUM)

- **Type:** Request;
- **From:** ParkServiceGUI;
- **To:** ParkManagerService;
- **Reply Message:** token
- **Payload:**
  - **SLOTNUM:** number between 0 and "number of slots";

- **Explanation:** From the requirements it is known that the user must send the slotnum previously received in order to enter the parking-area;

token(TOKEN)

- **Type:** Reply;
- **From:** ParkManagerService;
- **To:** ParkServiceGUI;
- **Request Message:** carenter
- **Payload:**
  - **TOKEN:** unique token relative to a couple car-parkingslot;

- **Explanation:** From the requirements it is known that after a user has parked the car in the indoor-area and has sent the slotnum, the service send back a token to pickup the car;

pickUp(TOKEN)

- **Type:** Request;
- **From:** ParkServiceGUI;
- **To:** ParkManagerService;
- **Reply Message:** canPickUp
- **Payload:**
  - **TOKEN:** unique token relative to a couple car-parkingslot;

- **Explanation:** From the requirements it is known that if the user wants to pickup the car must send the token received at the beginnig from the service;

canPickUp(X)

- **Type:** Reply;
- **From:** ParkManagerService;
- **To:** ParkServiceGUI;
- **Request Message:** pickUp
- **Payload:**
  - **X:** generic informational message;

- **Explanation:** From the requirements it is known that the service will notify the user if he/her can pickup the car at the outdoor-area;

canEnterCar(X)

- **Type:** Dispatch;
- **From:** ParkManagerService;
- **To:** ParkServiceGUI;
- **Payload:**
  - **X:** generic informational message;

- **Explanation:** From the requirement analysis it is known that the service will notify the user when he can move car to the indoor;

Timer for DTFREE and ITOCC

From the requirements are needed two timers for handling the entering/exiting of the cars (DTFREE and ITOCC). If we directly let the ParkManagerService manage the count-down it would be forced to stop until the end of the count-down; so it is necessary to introduce two active entities as timers that will notify the ParkManagerService at the end of the count-down. Since we are using the QAK

system they will be modelled as first level actor that we call ItoccCounter and DtfreeCounter.

Message Type Analysis

startItoccCounter(X)

- **Type:** Dispatch;
- **From:** ParkManagerService;
- **To:** ItoccCounter;
- **Payload:**
  - **X:** generic informational message;
- **Explanation:** The message is a *command* to start the count;

startDtfreeCounter(X)

- **Type:** Dispatch;
- **From:** ParkManagerService;
- **To:** DtfreeCounter;
- **Payload:**
  - **X:** generic informational message;
- **Explanation:** The message is a *command* to start the count;

itoccReached

- **Type:** Dispatch/Event;
- **From:** ItoccCounter;
- **To:** ? ;
- **Payload:** ? ;
- **Explanation:** The projectist has to decide if this message is a Dispatch or an Event:
  - if he choices it is a dispatch, then he also has to decide if the message can be send to the *ParkServiceGUI* or if it has to be sent to the *ParkManagerService* (but this could introduce a small overhead for the service);
  - if he choices it is an event then he also has to decide if the event can directly be handled by the *ParkServiceGUI* or if it has to be handled by the *ParkManagerService* (but this could introduce a small overhead for the service); this way is more reusable because of the actor who emits the event does not need to say who is recipient of the message.

dtfreeReached

- **Type:** Dispatch/Event;
- **From:** DtfreeCounter;
- **To:** ? ;
- **Payload:** ? ;
- **Explanation:** The projectist has to decide if this message is a Dispatch or an Event:
  - if he choices it is a dispatch, then he also has to decide if the message can be send to the *ParkServiceStatusGUI* or if it has to be sent to the *ParkManagerService* (but this could introduce a small overhead for the service);
  - if he choices it is an event then he also has to decide if the event can directly be handled by the *ParkServiceStatusGUI* or if it has to be handled by the *ParkManagerService* (but this could introduce a small overhead for the service); this way is more reusable because of the actor who emits the event does not need to say who is recipient of the message.

About devices

Transport Trolley

- **Code Avalability:** All the code to manage and comunicate with device is availlable because it is given bt the client;

Sonar

- **Code Avalability:** Only the c++ code for reading the hardware value is availlable. No interaction module is given by the client;
- **Usage from Requirement Analysis:**
  - Used by the system to check outdoor availlability;
  - Used as status information;

Thermometer

- **Code Avalability:** No code availlable;
- **Usage from Requirement Analysis:**
  - Its information it is used to start and stop the fan when temperature is above TMAX;
  - Used as status information (what is the temperature?);

Weight Sensor

- **Code Avalability:** No code availlable;
- **Usage from Requirement Analysis:**
  - Used by the system to check indoor availlability;
  - Used as status information;

Fan

- **Code Avalability:** No code availlable;
- **Usage from Requirement Analysis:**
  - Used by the system to decrease parking-area temperature;
  - Used as status information;

The executable model (QAK)

- ***parkmanagerservice_1.0.qak*** : a first very basic executable model;
- ***parkmanagerservice_2.0.qak*** : evolution of the previous model with counters and a **Mock State** in

order to realize automated test plan;

Test plans

To carry out the test plan, a **Mock State** was used. It is a shared Singleton class that allows to keep the status information inside it so that they can be modified and used during tests or by other classes. Two utility Java classes were also introduced: *TcpActorSpeaker* and *JCoapObserver* . TcpActorSpeaker deals with the management of sending TCP messages to the actors in order to incorporate a common behavior in a single reference class. JCoapObserver, on the other hand, is a naive observer that observes what happens to a certain actor using the coap protocol which is incorporated in Qak system.

First Test Plan

(Test made with Junit and Java in *FirstTestPlan.java* , see also the new executable model ready for test using Coap *parkmanagerservice_2.1.qak*)

testNotifyInterestNoSlotFree (alternative scenario of Notify Interest)

- **Tests:** a client wants to enter but no slots are available;
- **Conditions to pass:**
  - The SLOTNUM sent from the system is equal to 0;
  - The system comes back to its job;
  - The ITOCC counter must not have changed;

testNotifyInterestSlotFree (main scenario of Notify Interest)

- **Tests:** a client wants to enter and there is one slot free;
- **Conditions to pass:**
  - The SLOTNUM sent from the system is equal to the number of the parking-slot free;
  - The parking-slot must become reserved;
  - The system must send a confirmation message for parking the car;
  - The ITOCC counter must be started;

testCarEnterIndoorFree (main scenario of Car Enter)

- **Tests:** a client has received the SLOTNUM, the indoor is reserved for him then presses CARENTER;
- **Conditions to pass:**
  - The system says to the user that can enter and start the counter;
  - The system replies with token message and stopped the counter;
  - The parking-slot must become occupied;
  - The system comes back to its job;

testNotifyInterestSlotFreeButIndoorAlreadyEngaged (alternative scenario d of Car Enter)

- **Tests:** indoor-area is occupied and a client request to enter;
- **Conditions to pass:**
  - The system notifies user to wait for entering car;
  - The system comes back to its job;

testUserNotEnterCarInItocc (alternative scenario a of Car Enter)

- **Tests:** when a user is waiting for entering the car but itocc is reached;
- **Conditions to pass:**
  - The system says to the user that can enter;
  - The system comes back to its job;
  - The ITOCC count has reached the max value;
  - The parking-slot previously occupied is now free;
  - The system comes back to its job;

testSubmitTokenInvalid (alternative scenario a of Submit Token)

- **Tests:** a user send the token to pickup his car but it is invalid;
- **Conditions to pass:**
  - The system sends an error message;
  - The system comes back to its job;
  - The DTFREE count is not started;

testSubmitToken (main scenario of Sumbit Token)

- **Tests:** a user sends the correct token to pickup his car and the outdoor area is free;
- **Conditions to pass:**
  - The system has validated the request;
  - The outdoor is occupied;
  - The parking-slot is free;
  - The DTFREE count is started;
  - The system comes back to its job;

testSubmitTokenButOutdoorEngaged (alternative scenario b of Submit Token)

- **Tests:** a user sends the correct token to pickup his car and the outdoor area is already engaged;
- **Conditions to pass:**

- The system says the client to wait;
  - The system comes back to its job;

---

testUserNotPickUpInDtfree (alternative scenario a of Pickup Car)

- **Tests:** a user sends the correct token to pickup his car and the outdoor area is free, the user does not take the car and the dtfree counter is reached;
- **Conditions to pass:**
  - The system has validated the request;
  - The outdoor is occupied;
  - The parking-slot is free;
  - The DTFREE count is started;
  - The ITOCC count has reached the max value;
  - The outdorr is free;

Work Plan

1) Devices (12 hours)

Device software management:
- Sonar (3 hours);
- Weight sensor (3 hours);
- Fan (3 hours);
- Thermometer (3 hours),

2) ParkManagerService (3 days)

3) ParkingServiceGUI (2 days)

4) ParkServiceStatusGUI (2 days)

Project

Testing

Deployment

Maintenance

Automated Car Parking (Mattioli-Marchegiani)

mbot

By studentName email: student@studio.unibo.it