Workplan Automated Car-Parking- Sprint 1:

Requirements

The requirements given by the costumers Automated Car-Parking considered in the following SPRINT are:

- 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; A ***map*** of the parking area, represented as a grid of squares of side length ***RD***, is available in the file parkingMap0.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 ParkManagerService should perform the following tasks:

- ***moveToIn***: move the transport trolley from its current localtion to the INDOOR ;
- ***moveToSlotIn***: move the transport trolley from the INDOOR to the selected parking-slot;
- ***moveToHome***: if not already at home, move the transport trolley to its home location.
- ***moveToSlotOut***: move the transport trolley from its current location to the CARSLOTNUM/parking-slot ;
- ***moveToOut***: move the transport trolley to the OUTDOOR ;

The following sprint will focus to realize basic movements of the transport trolley.

Requirement analysis

Our interaction with the customer has made it clear what he means for:

- parking-area: an empty room showed in the map, where:

• "O" is the OUTDOOR-area; • "I" is the INDOOR-area ; • "r" is the home ; • "X" are the walls; • "s" are the slots.	|r , 0, 0, 0, 0, 0, I, X, |0, 0, s, s, 0, 0, 0, X, |0, 0, s, s, 0, 0, 0, X, |0, 0, s, s, 0, 0, 0, X, |0, 0, 0, 0, 0, 0, O, X, |X, X, X, X, X, X, X, X,

- home: an area where the robot is at in the beginning and where it goes when it hasn't any requests;
- INDOOR-area: the area in front of the parking INDOOR where is the weightsensor, the trasport trolley loads the car and the client leaves the car;
- OUTDOOR-area: the area in front of the parking OUTDOOR where is the OUTsonar, the trasport trolley leaves the car and the client takes his/her car;
- parking-slots: number of slots in the parking where the robot parks the car and the transport trolley couldn't pass on;
- current state: state of the trasport trolley (idle, working or stopped);
- fixed obstacles : any fixed element in the parking-aerea which the robot could collide with (e.g. a wall);

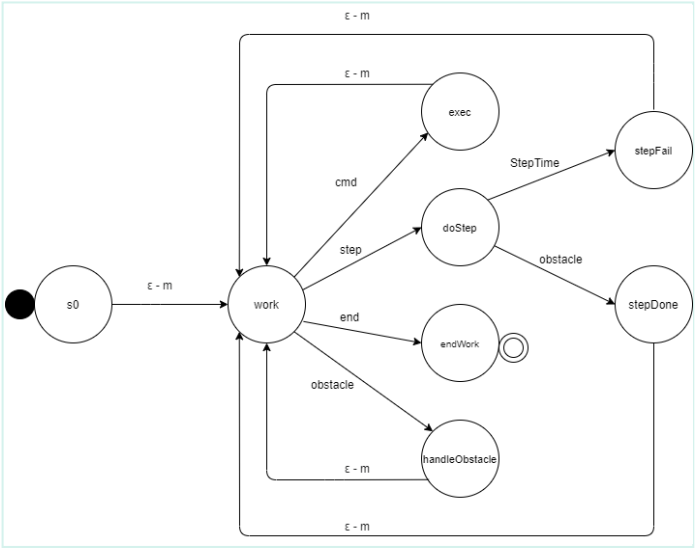
The DDRrobot

The model of the DDR-robot is related to the basicrobot given by the costumer: basicrobot2021.html.
The DDR-robot could be a VirtualRobot, Mbot or Nanobot:

- **virtual**:
  the virtual robot described in Virtual robot 2021, that can be launched by using the docker-compose file virtualRobotOnly2.0.yaml. The support is: virtualrobotSupport2021.kt
- **nano**:
  the self-made robot described in LabNanoRobot | The home-made basicrobot The support is: nanoSupport.kt
- **mbot**:
  the robot described in Mbot3030 | The reference ddr robot The support is: mbotSupport

The basicrobot

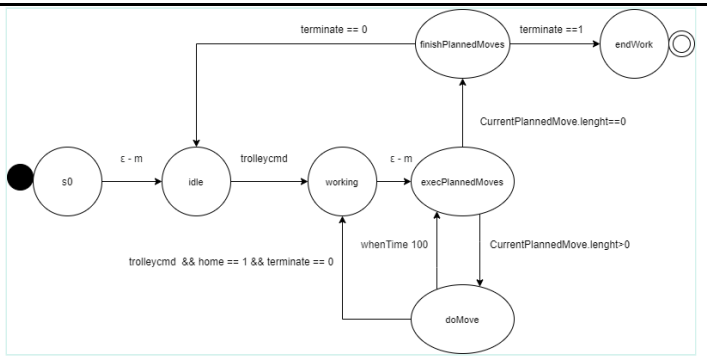
The basicrobot will use to commununicate between the DDR robot and the transport-trolley because it allows us to execute the robot movement commands in a 'technology-independent' way, with respect to the nature of the robot (virtual or real).
The configuration file basicrobotConfig.json, located in the repository it.unibo.sprint1, is used to respect the Virtualrobot nature by the basicrobot.



The transport-trolley

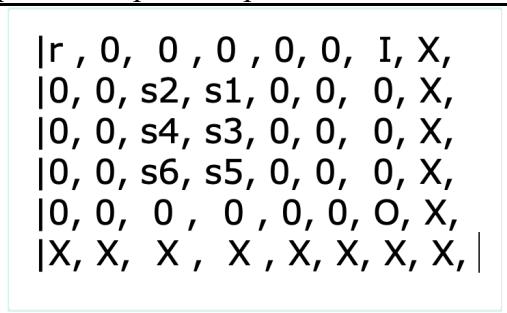
In our analysis the transport-trolley should represent the buisness logic of the DDR Robot. It will use the DDRrobot to execute its opertation that comes from the requirements of the application. In fact it should be able to generate a sequence of actions to reach the commands given by the ParkManagerService and exceute them sending the basic moves to the DDRrobot through the basic robot. It is a computational entity with a own autonomous flow of control so it will be modelled as QActor.

The trasport trolley should reach the goal place in a organizzed way: so the transport-trolley should planning (detection) the best sequence of actions and excecute them (actuation: e.g. sending messages to the basic robot).
The following file, given by the costumer, LabPlanner.html is able to planning the actions of a robot working in logical space using AI algorithms supporting.



Transport-trolley/Planner

The plannerUtil.kt is the support of the transport-rolley that is able to generate a plan of basic actions.
The parkingMap0.bin is the a binary file that represents the parking-area. It will be used by the planner to plan the path.

Because the transport-trolley shoud moves in a organizzed way the slots could be ordered in the following way so the first numbers will be near the INDOORarea and the transport troley will be faster to park the car. The plannerUtil.kt consider the following pair of number the respective places: • ("6","0") -> INDOORarea • ("6","4") -> OUTDOORarea • ("0","0") -> Home	

```
        run itunibo.planner.plannerUtil.initAI()
        println("&&&  trolley loads the parking map from the given file ...")
        run itunibo.planner.plannerUtil.loadRoomMap("parkingMap0.bin")
        //Set the parking area ...
        run itunibo.planner.plannerUtil.showMap()
    run itunibo.planner.plannerUtil.showCurrentRobotState()
```

After initializating the support the trasport trolley will generate the sequence of actions setting the cells goal with

```
        //for example run itunibo.planner.plannerUtil.planForGoal("6","0")
```

Comunication

Transport-trolley/Basicrobot

The Transport-trolley will send the command to the basicrobot in a asynchronous way to reduce the quantity of messeges and prevent the application from waiting for a response.	``` Dispatch cmd : cmd(MOVE) Context ctxparkingarea ip [host="localhost" port=8021] QActor transporttrolley context ctxparkingarea{ ... forward basicrobot -m cmd : cmd(w) ... } ```

The WebApplicationPms

The ParkManagerService should be an application server and, as the costumer requires, it should be used by the clients through the ParkServiceGui and the ParkServiceStatusGui on any devices (smartphone, personal computer, tablet, etc). Therefore in our analysis we define it as an web application that should offer a distribuited service of car-parking.

Problem analysis

The list of requirements for this sprint is:

- the transport-trolley carries the car to the parking space assigned by the ParkerManagerService;
- the transport-trolley carries the car to the the exit after having picked it up from the parking-slots;
- the transport-trolley mustn't pass over the parking-slots;
- the transport-trolley go back to its house if it hasn't requests;

To realize basic movements of the trasport trolley (moveToIn, moveToSlotIn, moveToHome, moveToSlotOut, moveToOut), the application should be able to:

- generate a sequence of elementary actions in order to reach the assigned places;
- provide a map of the environment to verified the transport-trolley's position;
- handle the transport-trolley positioning constraints inside the parking area;
- planning the transport-trolley's movements to move it in an organized way;
- communicate with the transport-trolley.

> To destroy the abstraction gap is possible to use QACTOR which is a modelling language and defines a work model of the system based on actors behavior.

Components

The ParkManagerService

The ParkManagerService, in this sprint, will be a part of the WebApplicationPms. The ParkManagerService will be modelled as a QActor to simulate one part of requirements: sent to the transport trolley commands with the informations where it should move to.

The transport-trolley

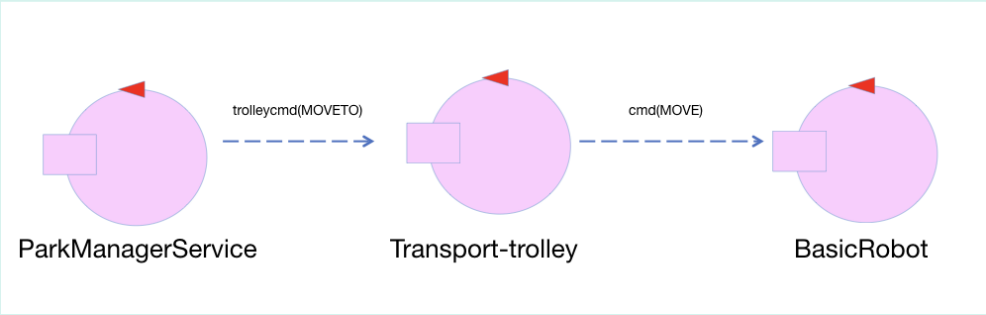
MoveToHome

The transport trolley when recives the command moveToHome should be able to excecute the command but also it shouldn't complete it if there is another reqeust from the ParkManagerService.

Load/Unload the car

The transport trolley when arrives near a SLOT/INDOORAREA/OUTDOORAREA it should load/unload the car so it should simulate the correct direction and the time to do this operation.

Components comunication



ParkManagerService/TransporTrolley

The ParkManagerService will send the command to the transport-trolley in a asynchronous way to reduce the quantity of messeges and prevent the application from waiting for a response.	`Dispatch cmd : cmd(MOVE)` `Context ctxparkingarea ip [host="localhost" port=8021]` `QActor parkmnagerservice context ctxparkingarea{` `...` `forward transporttrolley -m cmd : cmd(w)` `...` `}`

The first excecutable model is the following file: sprint1.qak

SPRINT REVIEW

This first analysis was produced some aspects that will be considerend in the next SPRINT.
The ParkManagerService should manage the reqeustes from the client and forward the command to the transport trolley. For example:

- the pair of commands moveToIn/moveToSlotIn or moveToSlotOut/moveToOut should be excecuted together by the transport-trolley after the properly checking by the ParkManagerService;
- if a slotnum is free it should be verified by the ParkManagerService. The slotnum will be use by the transport-trolley to set the slot where it should go.

Testplans

Testplan 1: the testplan should check the correspondence of the movements between the transport-trolley and the basicrobot.
The test will be done by comparing two strings:

- path: are the commands to reach the goal;
- result: are the real commands sendingo to the DDR robot from the basic robot.

At the end of the execution the two strings should be equals.
The code is the following file: TestPlan1.kt

Testplan 2: the testplan should check that the robot stopping the movements MoveToHome when it recives another requste (e.g. MoveToSlotIn).
The test will be done by comparing two strings:

- path: are the commands that the basicRobot will send to the DDR robot if the moviment MoveToHome isn't stopped.
- result: are the real commands sendingo to the DDR robot from the basic robot.

At the end of the execution the two strings should be differents.
The code is the following file: TestPlan2.kt

To realize an authomatized test we will use JUNIT.

Project

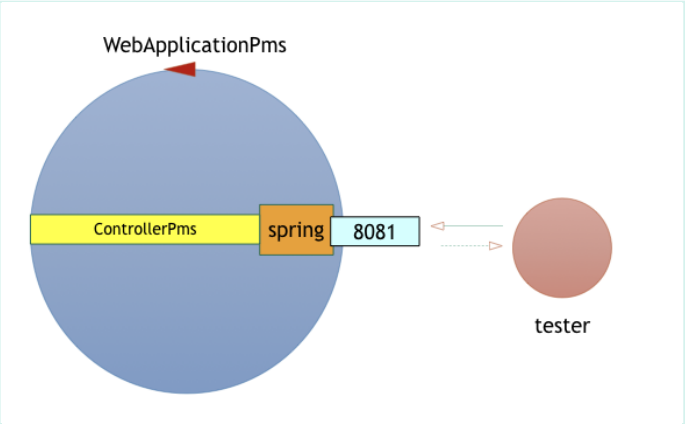
WebApplicationPms

The WebApplicationPms will be a system of actors and other components that implements the required automation functions. It is a distributed service accessible via HTTP-REST:

```
http://localhost:8081
```

We used the REST architecture because it allows each component to be identified as an individually addressable resource. Furthermore, using the exact same standard naming scheme as all other web resources allows you to seamlessly integrate Things and their properties into the web because their functions, data or sensors can be linked, shared, bookmarked and used as anything else on the web.
In this sprint there is a first version of the Web Application, whose only purpose, at the moment, is to send commands to the transport-trolley.
The communication between our system and an external component was simulated thanks to an object called a tester: Tester.kt

The WebApplication is in the following folder:
it.unibo.WebApplicationPms

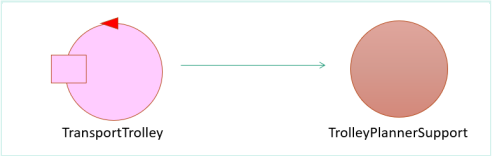


ControllerPms

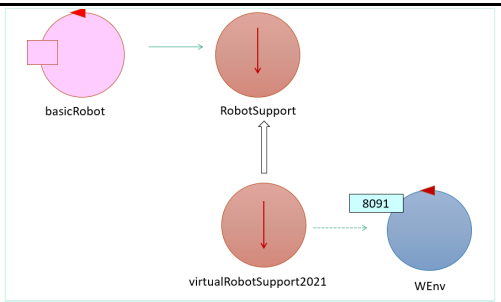
The ControllerPMS represents the WebApplication controller that defines the set of Representational State Transfer application programming interface (API RESTful).
This API has the task of creating a standard communication so that external components can communicate with our application and without knowing its implementation aspects.
The ControllerPms is the following file: ControllerPms.kt

```
POST /movetrolley
INPUT: move of transportTrolley (movetToIn)
OUTPUT: result of the move
```

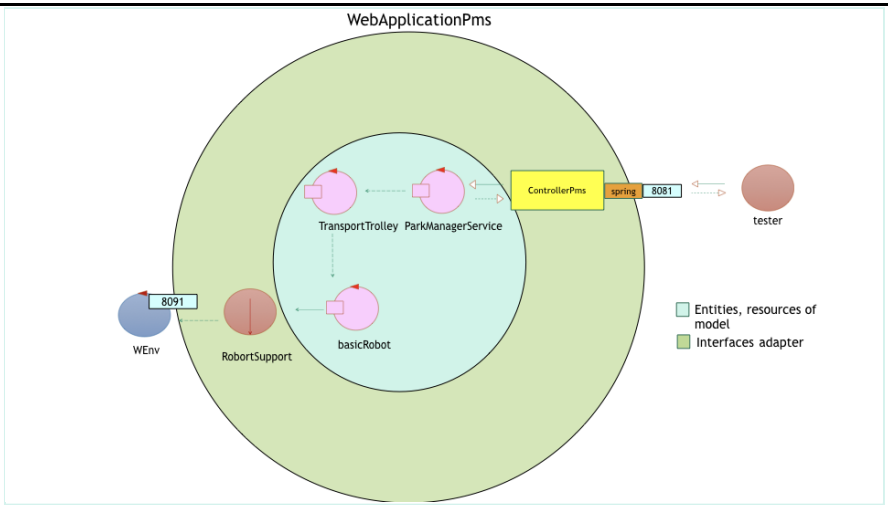
The transport-trolley

To sadisfy the single responsibility principle (SRP) it was decided to create a support for the transporTrolley. The tasks of the TrolleyPlannerSupport.kt are to plan the greatest path and update the map, thus the transporttrolley will have the only responsibility of sending the commands to the basicrobot.	

The basicrobot

The basicrobot, as saied in analysis requirements phase, interacts with the real robot throught the object robotSupport.kt It is the adatper fo the devices described in the configuration file : basicrobotConfig.json In this project it is used a virtual robot that is inside the vitual evirorment WEnv. The messages will be sent to the websoket (port:8091).	

Logical Architecture



The qactor excecutable model after project phase is the following file: sprint1_project.qak

Testing

After carrying out the design phase, the movements of the transporTrolley were tested again, involving the new components introduced in the tests.
In particular, for each possible command (moveToIn / MoveToSlotIn / moveToSlotOut / MoveToOut / MoveToHome) the correspondence of the movements between the transport trolley and basicrobot was checked.
The tests will be done by comparing two strings:

- path: are the commands to reach the goal;
- result: are the real commands sendingo to the DDR robot from the basic robot.

At the end of the execution the two strings should be equals.

The code is the following file: Testing.kt

Deployment

To run the ParkManagerService system, the WebApplicationPms file and the tester must also be launched. The project is located in the repository:
https://github.com/noemival/ParkManagerService_2021/tree/main/it.unibo.parkManagerService

By studentName email: antonio.iacobelli@studio.unibo.it

By studentName email: noemi.valentini5@studio.unibo.it