LABORATORIO DI INGEGNERIA DEI SISTEMI SOFTWARE

Introduction

This case-study starts to deal with the design and development of proactive/reactive software systems that use aynchronous exchange of information.

Requirements

Design and build a software system that allow the robot described in ***VirtualRobot2021.html*** to exibit the following behaviour:

- the robot lives in a closed environment, delimited by walls that includes one or more devices (e.g. sonar) able to detect its presence;
- the robot has a **den** for refuge, located near a wall;
- the robot works as an *explorer of the environment*. Starting from its **den**, the robot moves (either randomly or - preferably - in a more organized way) with the aim to find the fixed obstacles around the **den**. The presence of mobile obstacles is (at the moment) excluded;
- since the robot is *'cautious'*, it returns immediately to the **den** as soon as it finds an obstacle. Optionally, it should also return to the **den** when a sonar detects its presence;
- the robot should remember the position of the obstacles found, by creating a sort of 'mental map' of the environment.

Requirement analysis

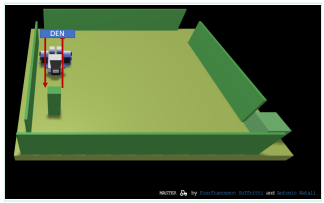
The **interaction with the client** made it clear that he or she associates the following meanings with nouns listed below:

- **closed enviroment**:any environment located indoors or outdoors bounded by walls.
- **sonar**: device capable of detecting the presence of the robot
- **robot**: a device capable of moving by receiving commands via network, as reported in VirtualRobot2021.html
- **den**:represents the starting position of the robot and the position to which the robot must return once an obstacle has been found..
- **obstacle**:any element within the environment that collides with the robot.
- **mental map**:structure that contains the position of the obstacles with which the robot has come into contact.

Regarding actions (verbs):

- **moves**: the robot can move randomly or in an organized way (preferable).

A user story

As a user, I place the robot in the den cell (facing south) and then activate a system that sends movement commands to the robot (via wifi network). As a user I cannot interrupt the execution: the system must terminate autonomously, once the task has been performed. At the end of the execution of the system, I expect that the robot, once it encounters an obstacle, to return to the den, remembering where the obstacle is. As a user, I can reboot the system to find other obstacles by starting the robot from a different den.	

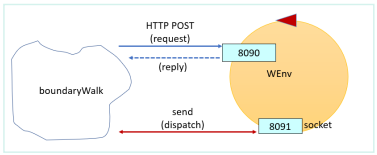
Problem analysis

Relevant aspects

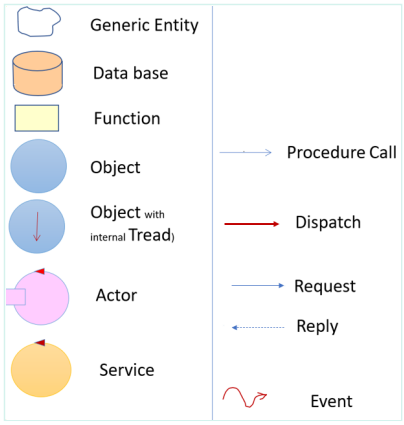
1. We need to build a **distributed system** consisting of two macro-components:
   - the (virtual) robot supplied by the client
   - our application (cautiousExplorer) that sends commands to the robot to satisfy the requirements
2. The robot can be controlled via network in two different ways, as described in VirtualRobot2021.html: commands
   - sending messages to the port 8090 with HTTP POST protocol
   - sending messages to the port 8091 using a websocket

3. Since there are numerous libraries in many programming languages that allow these command to be sent, no significant abstraction-gap is identified on the operational level.
4. This type of communication implies that both the application and the robot are connected to an internet network (wi-fi)

Logical architecture



Legend:



Given the information on how to use the robot, we can prefigure tthat the solution to the problem can be represented by following algorithm:

```
the robot starts from the den position, facing south (DOWN)
for 2 times:
        1) send  moveForward  commands to the robot until the robot encounters an obstacle or is detected by a sonar
        2) send for 2 times a  turnLeft  rotation command
```

TestPlan

Based on this analysis, we can define a testing strategy. To make the discussion easier, we introduce the following command abbreviations cril:

```
w : expresses the move {"robotmove":"moveForward", "time": 600}
s : expresses the move {"robotmove":"moveForward", "time ":300}
h : expresses the move {"robotmove":"moveForward", "time ":100}
l : expresses the move {"robotmove":"turnLeft ",   "time ":300}
r : expresses the move {"robotmove":"turnRight",   "time ":300}
```

The **testing strategy** identified consists in creating a **map of the territory** so as to know the path taken by the robot. By setting the **time** for the **w, s** moves to obtain **robot-unit** moves, we incrementally build the map after each move. Doing so, one can know where the robot is after each command and therefore, know the cell in which it has encountered an obstacle or it was detected by the sonar, and also we can check if the robot after finding an obstacle or after being detected by the sonar it returned to the den, for example:

- **1** represents the 'cells' traveled by the robot
- **r** represents the robot
- **X** represents the 'cell' in which the obstacle encountered by the robot is located or in which the robot was detected by the sonar

```
r, 0, 0, 0, 0,
1, 0, 0, 0, 0,
1, 0, 0, 0, 0,
X, 0, 0, 0, 0,
0, 0, 0, 0, 0,
```

Project

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