Lab ISS | the project cautiousExplorer with Actors

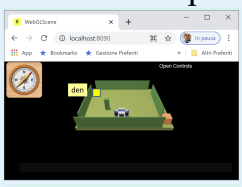
Introduction

This case-study deals with the design and development of proactive/reactive software systems based on the concept of Actor, as introduced in LabIss2021 | wshttp support with ActorBasicJava observers.

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 rectangular room, delimited by walls that includes one or more devices (e.g. sonar) able to detect the presence of obstacles, including the robot itself;
- the robot has a **den** for refuge, located in the position shown in the picture



- the robot works as an *explorer of the room*. Starting from its **den**, the goal of the robot is to create a map of the room that records the position of the fixed obstacles. The presence of mobile obstacles in the room is (at the moment) excluded;
- since the robot is *'cautious'*, it returns immediately to the **den** as soon as it finds an obstacle. It also stops for a while (e.g. 2 seconds), when it 'feels' that the sonar has detected its presence.

Requirement analysis

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

- **room**: a conventional room, such as those found in all buildings.
- **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.
- **map**:structure that contains the position of the obstacles with which the robot has come into contact.

- **cautious** when the robot encounters an obstacle it returns to the den following the path taken.
- **the robot works as an explorer of the room:** the robot does not know the room it is in, then it has to explore the environment in an organized way to identify obstacles and build a map
- **feels:**the robot receives a message from the sonar

A user story

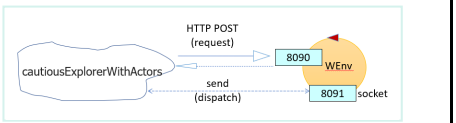
As a user, I place the robot in the den cell (facing south) and then activate a system that sends movement commands to the robot (via wifi network).
As a user I cannot interrupt the execution: the system must terminate autonomously, once the task has been performed.
As a user, I expect the robot to return to the lair after encountering an obstacle.
As a user, I expect that the robot stops for a while(e.g 2 seconds), when the sonar has detected its presence.
At the end of the execution of the system, I expect that the robot found all the obstacles and built a map of the room.

Problem analysis

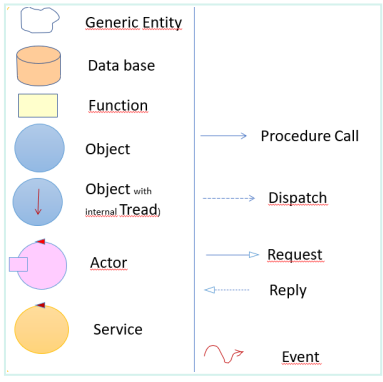
Relevant aspects

1. We need to build a **distributed system** consisting of two macro-components:
   - the (virtual) robot supplied by the client
   - our application (cautiousExplorerWithActors) that sends commands to the robot to satisfy the requirements

2. A first logical architecture:



Legend:



3. 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

Given the presence of sonar we need to develop a proactive/reactive software system. To create a system of this type it is better to use a web socket because the application does

not have to wait for a response.

4. From the interaction with the client, the following priority list of requirements emerged:
   - 1) the robot must return to the lair after colliding with an obstacle.
   - 2) the robot stops if detected by the sonar
   - 3) creation of the room map

5. 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.

6. We estimate that the first prototype of the application should be able to be built in 6 hours(at most).

7. The language that we will use to talk with out 'logical robot' is defined by the following grammar rule:

   ```
   ARIL ::= w | s | l | r | h
   ```

   Morover, if we assume here that the 'logical robot' can be included in a circle of diamater of length **DR**, the meaning of the aril commands can be set as follows:

   ```
   w : means 'go forward',  so to cover a length equals to DR
   s : means 'go backward', so to cover a length equals to DR
   h : means 'stop moving'
   l : means 'turn left of 90'
   r : means 'turn right of 90'
   ```

   This language will be used because it is more suitable for creating a map indicating the path of the robot

TestPlan

The **testing strategy** consists in building an incremental map of the territory in order to know the path taken by the robot.. Doing so, one can know where the robot is after each command and therefore, know the cell in which it has encountered an obstacle and also we can check if the robot after finding an obstacle it returned to the den. For example, assuming:

- **1** the 'cells' traveled by the robot
- **r** the robot
- **X** the 'cell' in which the obstacle encountered by the robot is located

when the robot encounters an obstacle we will have a map similar to the following:

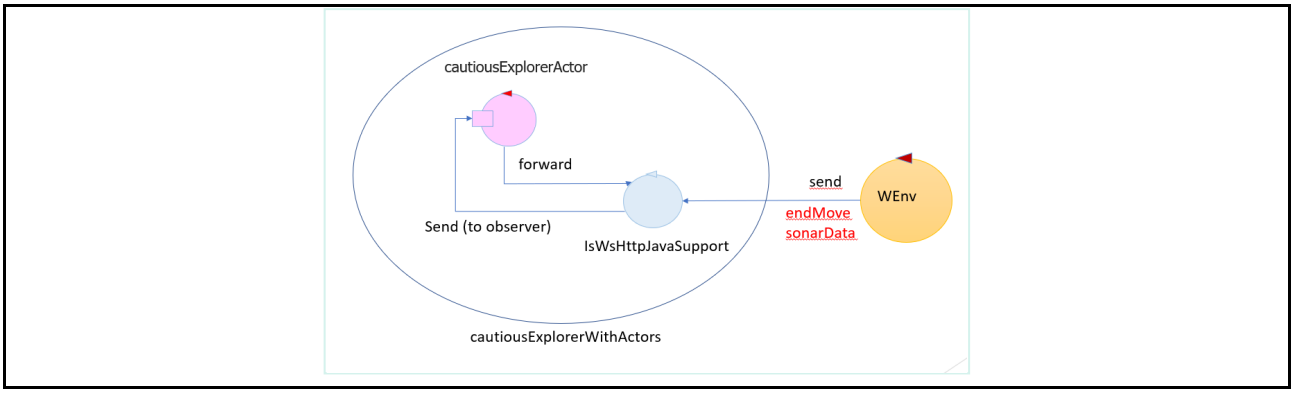
```
r, 0, 0, 0, 0,
1, 0, 0, 0, 0,
1, 0, 0, 0, 0,
1, 0, 0, 0, 0,
1, 0, 0, 0, 0,
X
```

while at the end of the application we will have a map of the following type:

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

Project

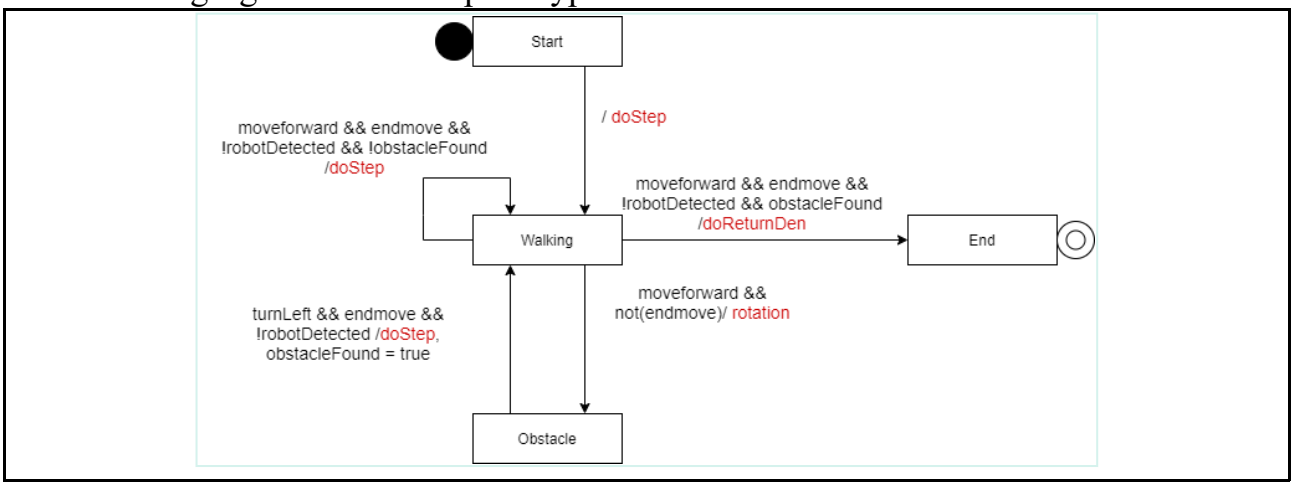
The cautiousExplorerWithActors application is a Java program, shown in the following figure:



A first prototype has been developed that satisfies the first two requirements:

- when the robot encounters an obstacle it returns to the den
- when detected by the sonar it stops for 1.5 seconds.

The following figure shows the prototype state machine:



Code: ***CautiousExplorerActor.java***

Testing

Deployment

The application can be found in my git: https://github.com/ant12-I/iacobelliAntonio/tree/master/it.unibo.cautiousExplorerWithActors

Maintenance

By studentName email:

antonio.iacobelli@studio.unibo.it