Lab ISS | the project cautiousExplorer

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.

Delivery

The customer requires to receive the completion of the analysis (of the requirments and of the problem) by **Friday 12 March**. Hopefully, he/she expects to receive also (in the same document) some detail about the project.
The name of the file (in pdf) should be:

```
cognome_nome_ce.pdf
```

Requirement analysis

Our **interaction with the custom** ha clarified that the customer intends:

- for **room**: a conventional (rectangular) room of an house;
- for **closed environment**: the room has a perimeter, that is physically delimited by solid **walls**;

- for **robot**: a device able to execute move commands sent over the network, as described in the document *VirtualRobot2021.html* provided by the customer;
- for **den**: starting point of the robot , it could be everywhere inside the room.
- for **obstacle**: something that do not allows robot to move through, walls are thinked as obstacle.
- for **return**: ability to come back to den.

The customer does not impose any requirement on the programming language used to develop the application.

Problem analysis

We highlight that:

1. In the VirtualRobot2021.html: commands the customer states that the robot can receive move commands in this way:
   - by sending messages to the port 8090 using **HTTP POST**
2. With respect to the technological level, there are many libraries in many programming languages that support the required protocols.

However, the problem does introduce an **abstraction gap at the conceptual level**, since **the required logical interaction** is always a ***request-response***, **regardless of the technology** used to implement the interaction with the robot.

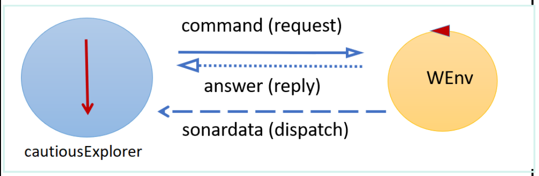
3. The Exploration Strategy is vital to detect object.At this point random approch is exclude , better focus on a exaustive predictable approch. Too close to call better one.
4. The need to return to den open the door to multiple solution. It's possible to use : stack , oriented graph or tree structure. Too close to call the best way.
5. How we know robot come back correctly to den? At this point it's not possible give an answer.

Logical architecture

We nust design and build a **distributed system** with two software macro-components:

1. the **VirtualRobot**, given by the customer
2. our **cautiousExplorer** application that interacts with the robot with a *request-response* pattern

A first scheme of the logical architecture of the systems can be defined as shown in the figure (for the meaning of the symbols, see the legenda)



We observe that:

- The specification of the exact 'nature' of our ***cautiousExplorer*** software is left to the designer. However, we have enought experience with previous projects that will be a java application..
- To make our ***cautiousExplorer*** software **as much as possibile independent** from the undelying communication protocols, the designer could make reference to proper ***design pattern***, e.g. **Adapter**, **Bridge**, **Facade**.

- In order to define **what the robot has to do** software must the meet the requirements:

```
We must define MemoryPath class related to the ability to remember the path of the robot.
The robot start in the den position with known orientation(or direction)
We must define an Exploration Strategy , and a ExplorationStrategy class .
```

Test plans

To check that the application fulfills the requirements, we set an exploration strategy and after a collision the robot come back. At this point is not possible to write a test plan.

Project

Testing

Deployment

Maintenance

By studentName email: ugo.marchesini@studio.unibo.it