LAB ISS | Cautious Explorer

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

- **robot**: a device able to execute move commands sent over the network, as described in the document VirtualRobot2021.html provided by the customer;
- **lives**: the robot is placed in a closed environment, such environment is delimited by 4 walls. It can move everywhere inside the environment.
- **walls**: the perimeter/boundary of the room.
- **devices**: something in the walls that detect the presence of the robot.
- **around**:
- **den**: the Home cell of the robot i.e., the cell where the robot will start to move. Such cell must be near a wall.
- **obstacle**: an obstacle placed inside the environment. There will be more obstacles inside the environment.

- **"mental map"**: the robot must save somehow and somewhere the information about the position of the obstacles in the map.

Verifying expected results (Test Plans)

We must be able to verify that the robot explores the map and finds an obstacle and memorizes them somehow and somewhere. Each time it finds an obstacle the return to the den is mandatory. In order to check these features we could memorize a list of string, one listitem for each obstacle found and where we memorize the moves done to reach the obstacle:

```
Test Test
```

Problem analysis

We highlight that:

1. In the VirtualRobot2021.html: commands the customer states that the robot can receive move commands in two different ways:
   - by sending messages to the port 8090 using HTTP POST
   - by sending messages to the port 8091 using a websocket
2. With respect to the technological level, there are many libraries in many programming languages that support the required protcols.

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. (Optionally) The robot must return to the den each time a sonar detects its presence, this logical operation is an event.
A first prototype can be done in two working days (at most).

Si possono far vedere anche più strade e evidenziare il modo migliore

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. It also receives events of detection of the robot

Since the requirements are also about the WEnv that generates an event when a device in the environment detects the robot, the protocol that best fit for this case is websocket -- the server can dispatch messages to the client without waiting for the client to send a request.

Test plans

Project

Scrivere che magari alcune parti sono già state sviluppate in boundarywalk e quindi che questi componenti verranno riutilizzati da noi. Raffinamento di una problematica o di una tematica che nasceva dai requisiti. Zoom dentro ciascun componente per fare vedere la architettura di progetto.

Testing

Deployment

Maintenance

By Andrea Zanni email: andrea.zanni8@studio.unibo.it