Lab ISS | iss0

ISS-lectures site

Introduction

Tenendo conto che: **there is no code without a project, no project without problem analysis and no problem without requirements** si riporta qui una possibile impostazioni dei primi passi di sviluppo per il progetto denominato **iss0** tenendo conto delle inidcazioni fornite nel template LABORATORIO DI INGEGNERIA DEI SISTEMI SOFTWARE .

In questo lavoro riporteremo anche alcune **considerazioni utili a comprendere alcuni aspetti fondamentali del lavoro dell'ingegnere del software**.

Requirements

Design and build a software system that leads a robot to walk along the boundary of a empty rectangular room.

Requirement analysis

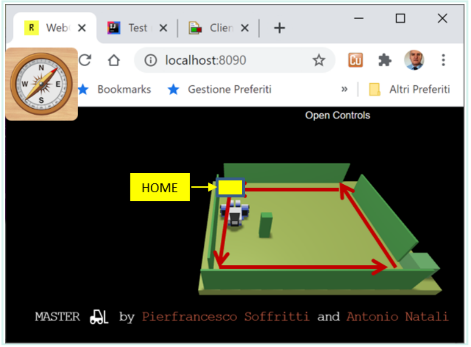
The **interaction with the client** made it clear that he associates the following meaning with nouns:

- **room**: a conventional room as found in all buildings
- **boundary**: perimeter of the room physically bounded by solid walls (**walls**)
- **robot**: a device capable of moving by receiving commands by the network as reported in **VirtualRobot2021.html**.

For the actions (verbs):

- **walk**: the robot must move forward hugging the walls of the room.

A first user story

As a user I place the robot in the HOME 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 has carried out (only once) the path shown in the figure to the side.	 Web Test Clien localhost:8090 Bookmarks Gestione Preferiti Altri Preferiti Open Controls HOME MASTER by Pierfrancesco Soffritti and Antonio Natali

Verification of expected results (Test plans)

It is necessary to verify that the path taken by the robot is the one expected.	The verification of the congruence of the path must be carried out via software without the need for intervention of a human user.

Problem analysis

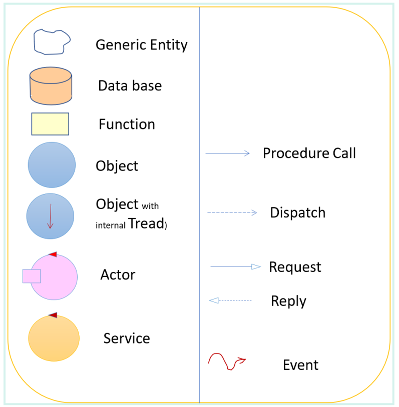
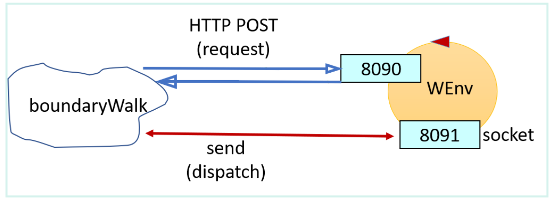
Relevant aspects

1. It is a question of creating a **distributed system** consisting of two macro-components:
   - the (virtual) robot supplied by the client
   - our application (**boundaryWalk**) that sends commands to the robot in order to meet the requirements
2. The robot can be controlled via the network in two different ways, as described in VirtualRobot2021.html: commands :
   - sending messages to port 8090 with HTTP POST
   - by sending messages to port 8091 using a websocket
3. Since there are numerous libraries in many programming languages that allow the sending of these commands no operationally significant **abstraction-gap** is identified.

However, there is a **abstraction gap on a conceptual level** : there are two possible technologies to communicate with the robot and each requires appropriate supports. However conceptually it is always about setting one **request-response interaction scheme** .

4. We estimate that a first prototype of the application should be able to be built in one working day (at most).

Logical architecture

Legend:  Generic Entity Data base Function Object Object with internal Tread) Actor Service Procedure Call Dispatch Request Reply Event	
 HTTP POST (request) 8090 WEnv boundaryWalk 8091 socket send (dispatch)	The exact nature of the **boundaryWalk** component will be defined in the design phase. Regarding the interaction we can say that: • the use of the HTTP protocol seems completely adequate, at least in a first phase; • the use of the websocket may prove to be *more flexible* (as it allows you to receive **information emitted by WEnv in a 'spontaneous' way**) and *more efficient* (as it reduces the protocol hierarchy).

Problems identified

1. **Interaction abstraction**

   The software system should be made as independent as possible from the communication protocol used for the interaction with WEnv.
   The designer could use some **design pattern** to make the application layer as **invariant** as possible with respect to the communication protocol which could be selected in a **configuration phase** of the system.

2. **Testing, but not only**

   The goal of defining an automatable TestPlan actually introduces a **new requirement** : make the application observable or better make its effects verifiable by a machine.
   In particular it is not enough to induce the robot to move; in the absence of cameras and / or special environmental sensors one way to proceed is to ***infer knowledge*** the commands sent and the responses obtained by WENV and **explicitly represent** that knowledge.

   We could say that it is necessary to create within the application a sort of ***'mind map'*** capable of providing a model of the robot's actions on which a testing program can 'reason' to carry out the verifications.
   Obviously such an approach does not ensure that the map matches to the real situation but this actually corresponds to what also happens to all sentient beings in the biological world which they operate on the basis of information obtained through the senses; in this case the 'sensors' would be the WEnv responses to the moves.

   More specifically given the information on how to use the robot we can prefigure that the solution to the problem consists of the following algorithm:

```
the robot starts from the HOME position, facing south (DOWN)
for 4 times:
        send moveForward commands to the robot until the robot hits the opposite wall
        and then I issue a turnLeft rotation command
```

Possible strategies for TestPlan: 'infer knowledge' from commands

Based on this analysis we can outline some different testing strategies. To make the discussion easier we introduce the following abbreviations related to cril commands:

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

Below is the testing strategy adopted:

1. **IPOTESI-TESTPLAN 3**: by setting the **time** for the **w, s** moves to obtain **robot-unit** moves we construct incremental a **territory map** after each move. This way you could know where the robot is after each command and then you could check the path taken at the end of the application; for example (**1** represents the 'cells' traveled by the robot). Moreover to be sure that the robot has covered the perimeter without any deviation it is necessary to compare the length of the perimeter expressed in robot units with the length of the initial perimeter of the room:

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

By Antonio Franzese antonio.franzese4@studio.unibo.it Github: https://github.com/Afranz1/Franzese-Antonio