LABORATORIO DI INGEGNERIA DEI SISTEMI SOFTWARE

Introduction

Our motto:
there is no code without a project, no project without problem analysis and no problem without requirements.

Requirements

Design and build a software system that makes a robot is able to walk along the boundary of a rectangular, empty room.

Requirement analysis

MAIN GOALS:

1. clarify the **meaning** of the *names* and of the *verbs* included in the requirement text given by the customer
2. (informally) define a first set of **functional TestPlans**

Interview

1. How is the robot? : the robot is virtual , specifications are defined in VirtualRobot2021
2. How robot detects walls? : robot has a front and rear sensor
3. How robot comunicate with client? : robot hosts a Node.js server and accepts commands and sends logs using json format.
4. How is the room? : the room has flat and not slippery floor
5. How can i locate the robot in then room ? : there's a sonar

Informally TestPlan

1. send command "run" and check logs that robot did the whole tour

2. send command "alive" and check logs that robot is on
3. send command "status" and check logs what robot is doing now (running ,last collision , etc)

Problem analysis

MAIN GOALS:

1. **Identify** the main problems involved by the requirements and the most appropriate (software) technologies to adopt
   The main problem is about detection of the current position/orientation of the robot. Libraries focused on communication and IO solves this issue.
2. **Evaluate** the **abstraction gap** and give a **quantitative measure of the effort/resources** necessary to build the system
   Abstration gap is about difference between communicaton/IO libraries and code that manage business logic about log analisys and specific json string contruction pattern
   In this moment i don't have enough information to evaluate the effort.
3. **Define** (a **model** of) the **logical architecture** of the system
   The entities of logic architecture are defined in Legenda.pptx
4. **Refine** the set of **functional TestPlan**
   TestPlans should be divided by architecture element and by functionality : for example for every command sent check the respose , include positive and negative test
5. (with reference to **SCRUM**) **Define** a (first) **product backlog** and a possibile set/sequence of **SPRINT**
   In this moment i don't have enough information to write a product backlog.
   We could divide SPRING sequence in :
   SPRING 1 : kick-off , build skeleton of the application (facade?)
   SPRING 2 : library for accepting command
   SPRING 3 : library for requests
   SPRING 4 : library for responses
   SPRING 5 : library for dispach

WARNING: expressions like *'we have chosen to ...'*, *'I decided ...'*, etc. are **forbidden** here. Rather, this section should include sentences like ***'this (aspect of the) problem implies that ...'*** or ***'the usage of this (legacy) component requires that ...'***, etc.

Test plans

MAIN GOALS:

1. with reference to the **logical architecture** of the system,

Write a program (e.g. by using **JUnit**) that defines the set of **functional TestPlans** that the software must satisfy.
In this moment i don't have enough information to write tests unit but enough to think about an hypothesis.
IPOTESI-TESTPLAN 3 Modified: impostando il **time** per le mosse **w, s** in modo da ottenere spostamenti di **unità-robot**, costruiamo in modo incrementale una **mappa del territorio** dopo ogni mossa. In questo modo si potrebbe sapere dove si trova il robot dopo ogni comando e quindi si potrebbe verificare al termine della applicazione il percorso effettuato; ad esempio (**2** rappresenta le 'celle' percorse dal robot,**1** rappresenta le 'celle' che saranno percorse dal robot , **0** le celle proibite):

```
Example 1 , tour is complete:

|w, w, w, w, w, X, w,
|X,sv, 2, 2, 2, 2, w,
|w, 2, 0, 0, 0, 2, w,
|w, 2, 0, 0, 0, 2, w,
|w, 2, 0, 0, 0, 2, w,
|w, 2, 2, 2, 2, 2, X,
|w, X, w, w, w, w, w,

Example 2 , still running:

|w, w, w, w, w, X, w,
|X, 2, 1, 1, 1, 1, w,
|w, 2, 0, 0, 0, 1, w,
|w, 2, 0, 0, 0, 1, w,
|w, 2, 0, 0, 0, 1, w,
|w, 2, 2, >, 1, 1, X,
|w, X, w, w, w, w, w,
```

Where : s = robot is stopped
v = heading down
> = heading right
< = heading left
^ = heading top
The robot start at the top left of the room heading to down.
When robot is moving show only arrow , when is stopped show s + arrow.
This choice allows to get more information about position,dinamyc status(run/stopped) and orientation of the robot , and where it collide with the wall

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