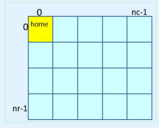
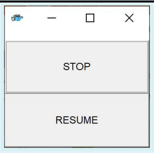
Lab ISS | the project resumableBoundaryWalker

Introduction

This case-study starts to deal with the design and development of proactive/reactive software systems which work under user-control.

Requirements

Design and build a software system (named from now on 'the application') that leads the robot described in *VirtualRobot2021.html* to walk along the boundary of a empty, rectangular room under user control.
More specifically, the **user story** can be summarized as follows:

the robot is initially located at the **HOME** position, as shown in the picture on the rigth	
the application presents to the user a **consoleGui** similar to that shown in the picture on the rigth	
when the user hits the button **RESUME** the robot **starts or continue to walk** along the boundary, while updating a **robot-moves history**;	
when the user hits the button **STOP** the robot stop its journey, waiting for another **RESUME** ;	
when the robot reachs its **HOME** again, the application *shows the robot-moves history* on the standard output device.	

Requirement analysis

During a meeting with the client the following definitions are clarified:

- **room**: a conventional room, as found in all buildings
- **boundary**: perimeter of the room, physically bounded by solid walls
- **robot**: a device capable of moving by receiving commands via the network, as reported in VirtualRobot2021.html.
- **robot-moves history**: set of moves that the robot performs displayed in string format output
- **consoleGUI**: a graphic interface used to send commands to the robot.
- **journey**: the path taken by the robot along the perimeter of the room.

Regarding actions (verbs):

- **walk**: the robot must move forward, hugging the walls of the room.
- **stop**: the robot does not continue the path.

Verification of expected results (Test plans)

It is necessary to check that the robot, in addition to following the expected path, moves when the	The verification of the congruence of the path and the correct execution of the

RESUME command is sent and stops when the STOP command is sent.	commands must be carried out through software, without the need for the intervention of a human user.

Problem analysis

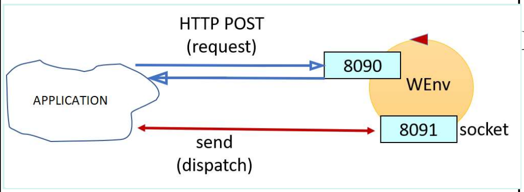
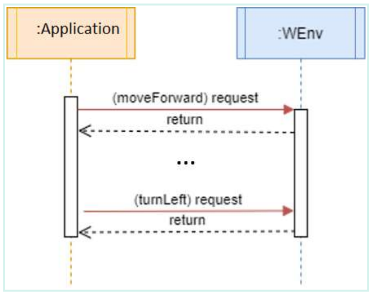
Relevant aspects

1. Create a distributed system consisting of two macro-components:
   - the (virtual) robot supplied by the client
   - our application which sends commands to the robot with a request-response pattern in order to satisfy 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 protocol
   - sending messages to port 8091 using a websocket
3. Since there are numerous libraries in many programming languages that allow the sending of such commands, no significant abstraction-gap is identified on an operational level.
4. We estimate that a first prototype of the application should be able to be built in 6 working hours.

The following resources could be usefully exploited to reduce the development time of a first prototype of the application:

1. The Consolegui.java (in project it.unibo.virtualrobotclient)
2. The RobotMovesInfo.java (in project it.unibo.virtualrobotclient)
3. The RobotInputController.java (in project it.unibo.virtualrobotclient)

Logical architecture

 HTTP POST (request) 8090 WEnv APPLICATION 8091 socket send (dispatch)	The exact nature of the 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 could prove to be more flexible (as it allows to receive information emitted by WEnv in a 'spontaneous' way) and more efficient (as it reduces the protocol hierarchy).
Another model (UML)  :Application :WEnv (moveForward) request return ... (turnLeft) request return	This model 'abstracts' from the technological details of the interaction.

About the robot-moves history

The robot walks along the boundary of the room.
We can build two possible different types of robot-moves history:

- a string that represents the robot path expressed as a sequence of moves. For example:
  wwwlwwwlwwwlwwwl
- the room is divided into cells of the size of the robot, in order to build a map as in the example:
  |r, 1, 1, 1, 1,
  |1, 0, 0, 0, 1,
  |1, 0, 0, 0, 1,
  |1, 1, 1, 1, 1,
  In this representation, we suppose that:
    - **r** means: cell occupied by the robot
    - **0** means: cell not explored
    - **1** means: cell explored

Problems identified

1. **Interaction abstraction**

   The specification of the exact 'nature' of our Application software is left to the designer. The software system should be made as independent as possible from the communication protocol used for the interaction with WEnv.

2. **Testing**

   We can prefigure that the solution of the problem consists of the following algorithm:

```
let us define emum direction {UP,DOWN,LEFT,RIGHT}
the robot starts from the HOME position, direction=DOWN when the user click RESUME
until the robot returns to the HOME:
        1) send to the robot the request to execute the command moveForward
        and continue to do it, until the robot hits the wall
        2) send the robot the request to perform the turnLeft


if the STOP command is received, the robot stops and restarts once the RESUME command is received,
executing the moveForward or the turnLeft based on the position of the robot
the robot returns to the HOME position and show the moves.
```

Test Plan

To check that the application fulfills the requirements, we could keep track of the moves done by the robot.

```

let us define String moves="";
for 4 times:
        1) send to the robot the request to execute the command moveForward;
        if the answer is 'true' append the symbol "w" to moves and continue to do 1);
        2) when the answer of the request becomes 'false',
        send to the robot the request to execute the command turnLeft and append the symbol "l" to moves
```

In this way, when the application terminates, the string moves should have the typical structure of a regular expression, that can be easily checked with a TestUnit software: moves: "w*lw*lw*lw*l" * : repetion N times(N>=0)

Project

Deployment

The deployment consists in the commit of the application on a project named **iss2021_resumablebw** of the MY GIT repository (**RRR**).

The final commit commit has done after **XXX** hours of work.

Maintenance

By Corsaro Lucrezia email: lucrezia.corsaro@studio.unibo.it