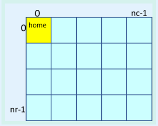
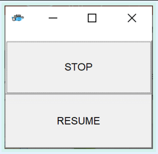
Lab ISS | the project resumableBoundaryWalker

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.	

Delivery

The customer **hopes to receive** a working prototype (written in Java) of the application by **Monday 22 March**. The name of this file (in pdf) should be:

```
cognome_nome_resumablebw.pdf
```

Requirement analysis

- Robot: the subject that will walk around the boundary of the room. Known robot's characteristics: *VirtualRobot2021.html*;

- Under user control: who uses the application must be able to start/stop/restart the robot when he/she wants;
- Home position: at the beginning, the robot must always be in the position shown in the first picture of the requirements;
- Console Gui: who uses the app must control the robot using a very simply graphical interface, only two buttons: resume (used also as a start) and stop;
- Robot-moves history: the robot must store all the moves that it has done. The history must represent only the moves of the current application session. The robot must store all the moves, also the "stop move", and the history must give some details about the move, e.g. the duration of the move or the space traveled;
- The application shows the robot-moves history: everytime the robot reaches the home position, the application must print the robot-moves history on the standard output;
- Standard output device: what the application uses to show the robot-moves history.

User story

At the beginning, the robot is in the home position looking to south. When the user hits the resume button, the robot starts to walk around the boundary of the room. If he/she wants, the user can hit the button stop and the robot will stop its walking. Then, the user can restart the robot walking clicking the resume button. Everytime that the robot reaches the home position, the application shows on the standard output device the moves history.

Informal Test Plan

- Don't stopping the robot, the application must always show an history that represent n times the same moves sequence;
- Clicking the stop button, the robot must be always stopped;
- Clicking the resume button while the robot is walking, the application must do nothing;

Problem analysis

The problem suggests a logical architecture composed by:

- a main service used to start/stop/restart the robot and to print the robot-moves history;
- a service used by the robot to receive commands from the previous service and to send back feedbacks to it.

The system can communicate with the robot using two ways:

- HTTP POST on port 8090
- websocket on port 8091

There is a conceptual abstraction gap because there are two possible ways of communication but both requires a request-response schema.

The system could use both the aril and the cril mode. To represent a move in a unique way, the aril mode is preferable beacuse in that mode a move consists of a space walked equals to the robot length.

The system must store the robot-moves history. It could be stored as a sort of list of the moves done, e.g. "wwwlwwwwlwwwl".

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

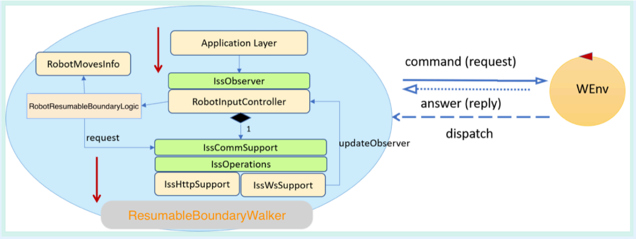
1. **Consolegui.java**: a useful artifact ables to show a simply gui;
2. **RobotMovesInfo.java**: an util to store and read the moves history;
3. **RobotInputController.java**: a controller that permits to say to the robot what it has to do and manages the asynchronous responses from the environment. It permits to use indifferently aril and cril.

The expected time required for the development of the application is (no more than) 6 hours.

Test plans

Project

The logical architecture presented during the problem analysis has been improved as follow:



Testing

Deployment

The deployment consists in the commit of the application on a project named **iss2021_resumablebw** of the my git repository (**https://github.com/danielecolautti/ColauttiDaniele_issLab2021**). The entry point to try the prototype is the main() method of iss2021_resumablebw.ResumableBoundaryWalker class.

The final commit commit has done after **6** hours of work.

Maintenance

By Daniele Colautti
email: daniele.colautti6@studio.unibo.it