

Lab ISS | the project resumableBoundaryWalker

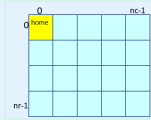
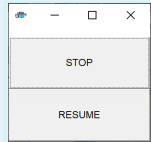
Introduction

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](https://www.virtualrobot2021.com/) 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 right	
the application presents to the user a consoleGui similar to that shown in the picture on the right	
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 reaches its HOME again, the application <i>shows the robot-moves history</i> on the standard output device.	

Requirement analysis

The **interaction with the client** made it clear that he or she associates the following meanings with nouns listed below:

- **room**: a conventional room, such as those found in all buildings
- **boundary**: the perimeter of the room, physically enclosed by solid walls (**walls**)
- **robot**: a device capable of moving by receiving commands via network, as reported in
- **walk**: the robot must move forward, along the walls of the room.
- **show robot-moves history**: print a map showing the path made by the robot

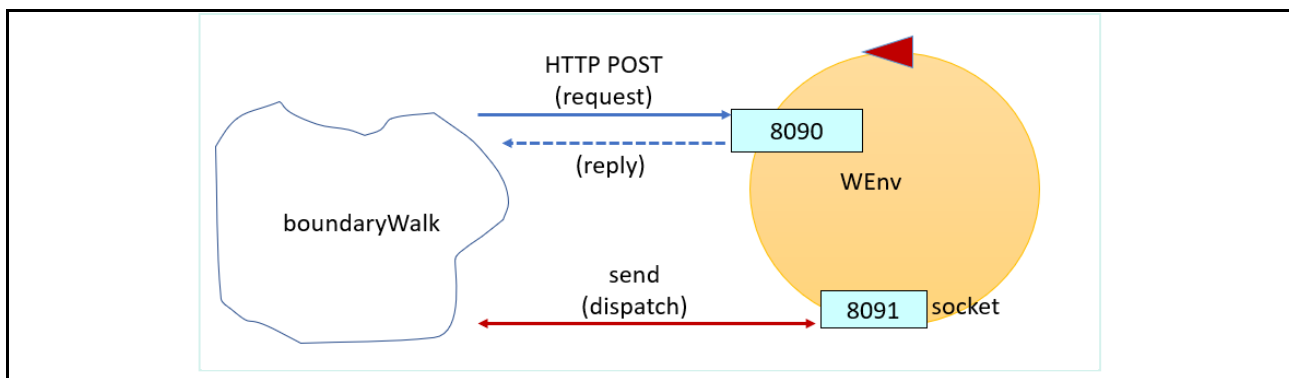
Problem analysis

Relevant aspects

1. We need to build a **distributed system** consisting of two macro-components:
 - the (virtual) robot supplied by the client
 - our application (resumableBoundaryWalker) that sends commands to the robot to satisfy the requirements
2. The robot can be controlled via network in two different ways, as described in [VirtualRobot2021.html: commands:](#)
 - sending messages to the port **8090** with HTTP POST protocol
 - sending messages to the port **8091** using a websocket

Since we have to develop a proactive / reactive software systems it is better to use a web socket because the application doesn't have to hang around waiting for a response.
3. Since there are numerous libraries in many programming languages that allow these command to be sent, no significant abstraction-gap is identified on the operational level.
4. We estimate that the first prototype of the application should be able to be built in 6 hours(at most).

Logical architecture



Legend: [legenda.pptx](#)

Test plans

The testing strategy can be summarized as follows:

TESTPLAN 1 :The testing strategy identified consists in creating a **map of the territory** so as to know the path taken by the robot. By setting the **time** for the **w, s** moves to obtain **robot-unit** moves, we incrementally build the map after each move. Doing so, one can know where the robot is after each command and therefore, check the path taken at the end of the application; for example:

- r means: cell occupied by the robot
- 0 means: cell not explored
- 1 means: cell explored
- X means: cell occupied by an obstacle

```
|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,
```

TESTPLAN 2 : the robot start in the HOME position, direction=DOWN. Define String commands="";

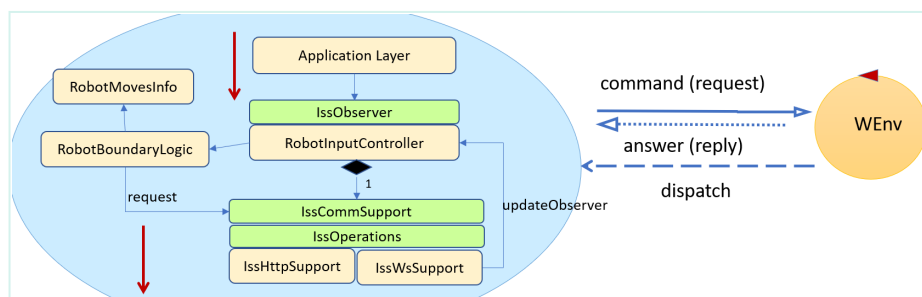
1. send to the robot the request to execute the command stop by GUI; if the answer is 'true' append the symbol "h" to commands.
2. send to the robot the request to execute the command resume by GUI; if the answer is 'true' append the symbol "w" to commands.

In this way, when the application terminates, the string commands should have the typical structure of a regular expression:

- **commands:** "hm"

Project

The boundaryWalk application is a conventional Java program, represented in the figure:



The language that we will use to talk with our 'logical robot' is defined by the following grammar rule:

ARIL ::= w | s | l | r | h

Moreover, if we assume here that the 'logical robot' can be included in a circle of diameter of length **DR**, the meaning of the aril commands can be set as follows:

```
w : means 'go forward', so to cover a length equals to DR
s : means 'go backward', so to cover a length equals to DR
h : means 'stop moving'
l : means 'turn left of 90'
r : means 'turn right of 90'
```

This language will be used because it is more suitable for creating a map indicating the path of the robot

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/ant12-I/iacobelliAntonio/tree/master"](https://github.com/ant12-I/iacobelliAntonio/tree/master)

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

Maintenance

By studentName email: antonio.iacobelli@studio.unibo.it

