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	emptyGridWithHome
the application presents to the user a **consoleGui** similar to that shown in the picture on the rigth	stopResumeConsole
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 **Monady 22 March**. The name of this file (in pdf) should be:

```
cognome_nome_resumablebw.pdf
```

Requirement analysis

The robot must stop with a limited latency, (under 1s)

Term	Meaning
Robot	The Object provided by the customer
HOME	Home cell of the robot
GUI	Graphical User Interface, should contain both the RESUME and STOP buttons
Room Moves History	A kind of map that remember robot moves
STOP	

Term	Meaning
RESUME	The robot must resume the walk with a limited latency without losing the current progress

Problem analysis

RobotBoundaryZooming

Key points: Asynchronous REQUEST and observed REPLY.

1. We start from an architecture based on the **Observer pattern**, with a controller that works as an observer of the communication support and takes care of handling information about obstacles
2. The system controller must now handle commands sent from the user console
3. The system controller should work as an observer of the user console
4. The user-command stop must stop the journey, not the current move
5. The boundary-walk logic must consider the fact that the journey could be stopped before doing the next journey action
6.

Robot language

ARIL is ok to use as the robot do not expect variable length obstacles while moving around. Movement by Robot Unit help the creation of the map.

Architecture

- **VirtualRobot**, given by the customer
- Our **ResumableBounaryWalker** application

The objects talk using a bidirectional message passing architecture.

Technology

Any technology that support the defined communication pattern with the robot is ok.

Test plans

The proposed test plan is so described:

1. The robot in his den is sent to walk the boundary
2. After a time T the resume command is issued
3. We expect to not see differences
4. After a time T the stop command is issued
5. We expect to stop receiving messages
6. After a time T the resume command is issued
7. We expect to start receiving messages
8. We expect that the map representation is the same as without the stop and resume

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)

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

Project

1. BoundaryWalkResumable.java create a GUI for the application, setup a WS communication using VirtualRobotWS.java, this is embedded in the ARIL Decorator VirtualRobotClientARIL.java
2. This object is passed to the BoundaryWalkController that act as a Supplier of a Future BoundaryWalkController.java
3. the GUI Controller called ResumeController ResumeController.java take the Supplier and make it generate the current Future
4. the stop and resume buttons are connected to this controller
5. The main frame is then shown and the walk boundary begin

Testing

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 Alessio Ferri email: alessio.ferri@studio.unibo.it