LABORATORIO DI INGEGNERIA DEI SISTEMI SOFTWARE

Introduction

Requirements

- Req 1: percorrere (una volta) il bordo perimetrale della stanza rappresentata ne La scena di WEnv

Requirement analysis

- virtualrobot: VirtualRobot23
- scena: la-scena-di-wenv

Problem analysis

il problema richiede di muovere il robot attraverso il perimetro tramite le chiamate http dopo un analisi dell'ambiente WEnv le interazioni possibili con il robot mediante le chiamate http sono le seguenti

{"robotmove":"turnLeft", "time":T}	il robot ruota di 90* verso sinistra nel tempo T
{"robotmove":"moveForward", "time":T}	il robot si muove in avanti a una velocita costante per il tempo T
{"robotmove":"turnRight", "time":T}	il robot ruota di 90* verso destra nel tempo T stabilito
{"robotmove":"moveBackward", "time":T}	il robot si muove all'indietro a una velocita costante per il tempo T stabilito

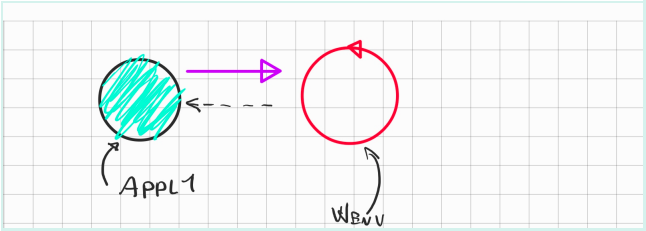
Il robot a ogni comando risponde con un oggetto JSON secondo la seguente sintassi

```
{"endmove":"RESULT", "move":"MINFO" }
```

dove RESULT indica l'esito dell'escuzione dell'operazione e MINFO fornisce le seguenti informazioni

MOVEID	l'identificativo in linguaggio cril della mossa eseguita
interrupted|collision	informazioni riguardati lo stato di eventuali eventi generatisi durante lo svolgimento dell'operazione, quali mossa interrotta e collisione avvenuta

ARCHITETTURA LOGICA



l'architettura è formata da due componenti:

- Appl1 è un POJO formato dall'applicazione che fa eseguire al robot il giro del perimetro
- il sistema WEnv rappresentato come server che riceve dal componente Appl1 le chiamate HTTP e risponde

OSSERVAZIONI

- il robot non specifica informazioni riguardanti la sua posizione, questo puo comportare difficoltà aggiuntive in fase di testing
- tramite le chiamate HTTP le informazioni riguardo a eventi e possibili vengono comunicate solo dopo il tempo stabilito tramite la richiesta, questo rende difficile l'intervenire su eventi in maniera tempestiva
- data la natura sincrona delle chiamate HTTP il testing della funzionalita è semplificato

VALUTAZIONE DELL' ABSTRACTION GAP

dall'architettura logica si determina che l'abstraction gap generato dal problema non è molto elevato,è possibile sfruttare librerie per gestire le chiamate HTTP e convertire le stringhe JSON all'occorrenza

ANALISI DEI COSTI

- per la progettazione e implementazione della funzionalità prevedo una tempistica di qualche ora, considerando la creazione di una struttura riusabile e il piu possibile protocol independent
- per il testing, data la natura delle comunicazioni con il robot prevedo un tempo piu lungo, vista l'impossibilità di conoscere la posizione esatta del robot

Test plans

Unit Test

il piano dei test deve essere scritto tenendo conto dei seguenti punti

- il robot, dopo l'esecuzione di BoundaryWalk() deve trovarsi nella stessa posizione da cui è partito
- il robot deve percorrere il perimetro dell'ambiente virtuale

il primo punto si puo verificare assicurandosi che se il robot si muove verso l'alto o verso sinistra si verifichi una collision

```
public void checkRobotIsComeBackToHome() {
```

```
RobotResponse[] response;
response = robot.doBoundaryWalk();
assertEquals(response.length, 4);
//check move to left generate collision
assertTrue(robot.moveLeft().contains("collision"));
//check move to top generate collision
assertTrue(robot.moveTop().contains("collision"));
}
```

per il secondo punto e necessario:

- misurare la dimensione della stanza
- che il metodo doBoundaryWalk() restituisca informazioni sulle metriche percorse

la struttura dati restituita dal robot è un array di POJO composto da:

- direzione in cui il robot si è mosso
- quantita di spazio percorsa

è necessario definire un unità di misura per determinare la dimensione della stanza, per consentire una facile implementazione si è scelta come unita di misura la distanza percorsa dal robot in un intervallo di tempo di un secondo, questa scelta è dipendente dalla velocita del robot e in caso di variazione della stessa sarà necessario rieffettuare le misure

```
public class RobotResponse { private int direction;
private int steps;
public RobotResponse() {
}
public int getDirection() {
return direction;
}
public void setDirection(int direction) {
this.direction = direction;
}
public int getSteps() {
return steps;
}
public void setSteps(int steps) {
this.steps = steps;
}
}
```

il testing puo essere effetuato paragonando le misure della stanza effettuate in precedenza con lo spazio percorso dal robot

```
public void checkRobotHasDoneBoundaryWalk() {
//assuming the method returns the number of steps in each direction
RobotResponse[] response;
response = robot.doBoundaryWalk();
assertEquals(response.length, 4);
assertEquals(response[0], dimensions[0]);
assertEquals(response[0], dimensions[1]);
assertEquals(response[0], dimensions[2]);
assertEquals(response[0], dimensions[3]);
```

}

Integration Test

eseguire in una macchina locale il componente di ricezione e l'ambiente virtuale e avvalendosi di un analizzatore della rete, verificare che le chiamate avvengano correttamente

System Test

eseguire in una macchina locale il componente di ricezione e l'ambiente virtuale e avvalendosi di NaiveGui.html, verificare che la funzionalita sia soddisfatta

Project

Testing

Deployment

Maintenance

By Matteo Longhi email: matteo.longhi5@studio.unibo.it,

GIT repo: https://github.com/carnivuth/iss_2023_matteo_longhi.git