LABORATORIO DI INGEGNERIA DEI SISTEMI SOFTWARE

Introduction

Requirements

Scrivere un programma Java che possa inviare al VirtualRobot (attraverso una **console remota**):

- un comando **start** per attivare il VirtualRobot in modo che percorra il bordo perimetrale della stanza rappresentata nella scena di WEnv
- un comando **stop** per bloccare l'azione del VirtualRobot.
- un comando di **resume** per riattivare l'azione del VirtualRobot.

Requirement analysis

- Per *VirtualRobot*, il committente intende: *virtual robot*
- Per *scena di WEnv*, il committente intende: *scena WEnv*

Dopo colloqui con il committente, possiamo dire che:

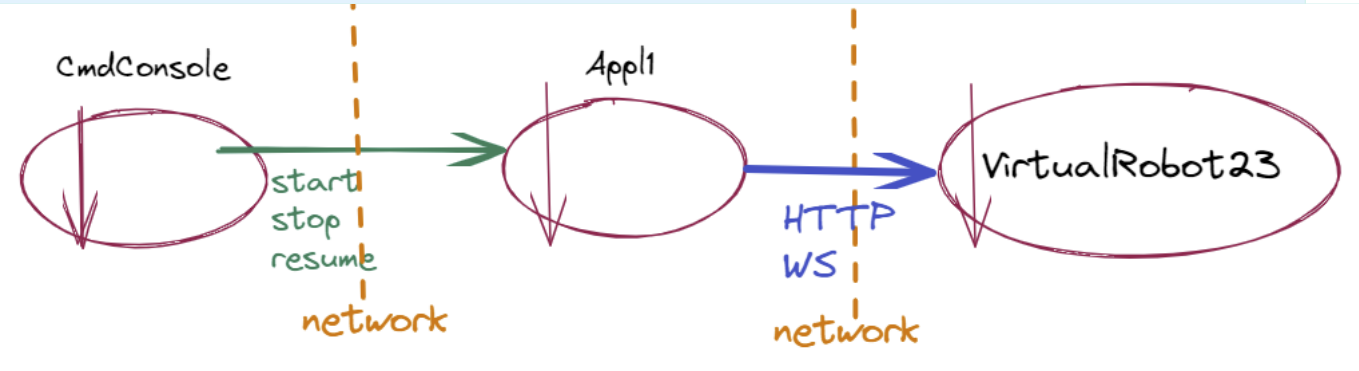
- osservando la documentazione del VirtualRobot, si deduce che può essere modellato come un **servizio**;
- il VirtualRobot23 (d'ora in poi denominato robot) si trova inizialmente nella situazione descritta come stato iniziale
- l'applicazione può inviare comandi al robot sia con **interazione sincrona** sia con **interazione asincrona**;
- un comando di **stop** inviato al robot che ha terminato il percorso non ha effetto;
- un comando di **resume** inviato al robot che ha terminato il percorso equivale a un comando di start;
- l'applicazione invia i comandi **stop/resume** sotto il controllo di un utente umano, attraverso la *console remota di comando*.

Il sistema è **distribuito** su diversi nodi computazionali: con *console remota* si intende il fatto che la console di comando (che denominiamo CmdConsole) è collocata su un computer diverso da quello in cui gira l'applicazione.

Problem analysis

- L'applicazione non agisce in modo autonomo, ma come conseguenza di comandi inviati da *CmdConsole*.
- Una volta ricevuto il comando **start**, l'applicazione presenta un comportamento al contempo *proattivo* (in quanto deve effettuare in modo autonomo un giro della stanza) e *reattivo* (in quanto deve essere capace di percepire ed eseguire i comandi **stop/resume**).

Dall'analisi dei requisiti e del problema, si può definire una prima architettura logica del sistema da realizzare:



Il sistema è logicamente formato da tre componenti:

- *VirtualRobot23*, che viene fornito dal committente.
  Il robot è un componente reattivo/proattivo: opera come un server e può ricevere comandi sia via rete tramita HTTP (**interazione sincrona**) sulla porta 8090 che tramite WebSocket sulla porta 8091 (**interazione asincrona**). Il server può anche inviare su WS messaggi che non siano risposte a comandi (**messaggio di stato**).
- *Appl1*, che dobbiamo realizzare
  È un ente attivo che attende un comando (**start/stop/resume**) da CmdConsole. Questi comandi possono essere inviati come messaggi via rete tramite un protocolllo e sono poi interpretati e convertiti in comandi al robot.
- *CmdConsole*, che dobbiamo realizzare
  Componente che si presenta come un ente attivo che interagisce con un utente umano e che invia comandi via rete ad Appl1.

Il core-business dell'applicazione è di competenza del componente *Appl1*, da progettare e realizzare per primo. La CmdConsole potrà essere progettata e realizzata in un secondo momento, prima in locale e poi nel distribuito.

Test plans

Dopo un ulteriore colloquio con il committente, si ritiene significativo sapere al termine dell'esecuzione dell'applicazione:

- se il robot è fermo in **HOME**
- se ha effettuato un (solo) giro lungo il perimetro della stanza.

Si è anche convenuto che in futuro potrebbe essere utile sapere:

- quanti comandi di stop/resume il robot ha ricevuto ed elaborato
- quale posizione il robot occupa nella stanza in un certo istante
- se il robot ha incontrato ostacoli mobili lungo il percorso
- ...

Azioni di testing pianificate

Concentrando al momento l'attenzione sul componente *Appl1*, possiamo pianificare le seguenti azioni di User Acceptance Test:

- **testStartNoStop**: dopo che *Appl1* ha ricevuto (da *CmdConsole*) il comando **start**, occorre verificare che il robot abbia iniziato la sua attività, cioè sia in moto e che, al termine, esso risulti fermo in HOME avendo completato un giro.
- **testStop**: dopo che *Appl1* ha ricevuto (da *CmdConsole*) il comando **stop**, occorre verificare che il robot sia fermo.
- **testResume**: dopo che *Appl1* ha ricevuto (da *CmdConsole*) il comando **resume**, occorre procedere come per *testStartNoStop*.

Project

Lo sviluppo dell'applicazione viene impostato in modo incrementale:

1. *Applicazione1HTTP*: l'applicazione interagisce con il VirtualRobot23 solo via HTTP
2. *Applicazione1HTTPeWS*: ammettiamo la possibilità di interagire con il VirtualRobot23 anche via WebSocket.

Applicazione1HTTP

La progettazione viene suddivisa nei seguenti SPRINT:

1. SPRINT1 *core problem* dell'applicazione: la fase *proattiva* in cui il robot percorre il perimetro della stanza *senza gestire i comandi stop/resume*.
2. **SPRINT2** estendiamo il sistema prodotto nello *SPRINT1* per realizzare la fase *reattiva* in cui il robot *gestisce i comandi stop/resume*.
3. **SPRINT3** estendiamo il sistema prodotto nello *SPRINT2* introducendo la *Console remota*.

Testing

Deployment

Maintenance

- By Letizia Mancini
- email: letizia.mancini3@studio.unibo.it
- GIT repo: https://github.com/llevtizia/issLab23-ManciniLetizia
- matricola: 0000926656