SPRINT 1

PROBLEM ANALYSIS

Assunzione sul numero di clienti

In questo primo sprint si fa l'assunzione di gestire solo **un cliente alla volta**, di conseguenza non bisogna pensare alle problematiche relative a:

- Gestione multipla dei tavoli
- Richieste multiple alla smartbell

Descrizione del Sistema

Si procede alla definizione più dettagliata degli **attori** che compongono il sistema:

- **Waiter**: gestisce la sala da the interagendo con il cliente e gli altri attori.
- **Smartbell**: interagisce direttamente con il cliente e si occupa di notificarne l'arrivo al waiter.
- **Barman**: interagisce esclusivamente con il waiter.
- **Client (esterno):** simula un cliente ed il suo ciclo di vita nel sistema.
- **Manager (esterno):** deve avere accesso al web server per consultare lo stato della stanza

Problematica di gestione dei tavoli

Per la gestione dei tavoli sono presenti diverse opzioni:

- Gestire i tavoli come variabili interne al waiter.
- Gestire i tavoli tramite una base di conoscenza, sempre gestita dal waiter.
- Gestire i tavoli tramite una base di conoscenza, gestita da un attore esterno

Tutte queste soluzioni risultano valide.

Per questo primo Sprint la cosa più vantaggiosa risulta essere la prima soluzione, perché si fa l'ipotesi di un solo cliente alla volta, di conseguenza bisogna tener conto solo di un tavolo.

Per Sprint futuri sarà meglio valutare le altre due opzioni, dato che risultano essere più scalabili e facilmente modificabili.

Waiter, walker, waiterwalker and basicrobot

Nonostante le assunzioni che semplificano il problema, è utile introdurre fin da subito una distinzione sui vari livelli relativi al waiter. In particolare, il waiter sarà partizionato in **più componenti (attori),** ognuno dei quali si occupa di un aspetto specifico:

- **Waiter**: si occupa della **logica di alto livello**, cioè di come debba comportarsi il waiter all'interno dell'applicazione.
- **Walker**: è il **middleware** fra il waiter e il waiterwalker, si occupa di prendere le richieste di movimento da parte del waiter e di inoltrarle al waiterwalker affinché vengano soddisfatte.
- **Waiterwalker**: si occupa di creare il **piano** per andare dal punto in cui si trova il robot alla destinazione. Il piano viene concepito come sequenza di azioni che vengono inviate al basicrobot per essere eseguite.
- **Basicrobot**: è il layer più basso, riceve in ingresso un comando e lo fa eseguire al virtual robot.

Il basicrobot e il waiterwalker sono stati forniti già completi e funzionanti:

- **Basicrobot**: riferimento a https://github.com/anatali/iss2020LabBo/tree/master/it.unibo.qak20.basicrobot
- **Waiterwalker**: https://github.com/anatali/iss2020LabBo/tree/master/it.unibo.qak20.domains

Tale suddivisione permette di avere un sistema altamente scalabile, così se è necessario modificare, per esempio, la posizione degli elementi nella stanza è necessario andare a modificare solo il walker, se bisogna modificare il robot fisico è necessario andare a modificare solo il basicrobot, e così via.

Problematica sulla rappresentazione dello spazio

Un'idea consiste nel rappresentare lo spazio tramite una **mappa**, con indicati su di essa i vari punti di interesse all'interno della sala.

Tale mappa dovrà servire al waiter per potersi muovere all'interno della sala, per questo motivo è opportuno rappresentare la mappa tramite una matrice di coordinate (x,y).

Ogni **cella** di questa matrice indica un punto all'interno della sala.

La nostra software house, come già indicato sopra, fornisce un basicrobot già fatto che ha una funzionalità molto utile, cioè il potersi muovere a **step.** Ogni step consiste nello spostamento da parte del robot di uno spazio pari alla dimensione del robot stesso.

Di conseguenza l'idea migliore per poter mappare la stanza consiste nel creare tale matrice dove ogni punto indica una posizione raggiungibile dal robot partendo dalla posizione iniziale (home) tramite un susseguirsi di step.

La mappa può essere creata sempre tramite un *progetto* fornito dalla nostra software house.

Così si può ottenere una mappa della sala come punti raggiungibili tramite step da parte del waiter.

Posizione corrente del waiter

Il robot inizia il suo ciclo sempre da una determinata posizione (la home), per poter conoscere la posizione è quindi sufficiente riferirsi alla mappa sopra descritta, in base ai movimenti eseguiti, si andrà a modificare la sua posizione.

Conoscendo il punto di partenza e le mosse eseguite è molto semplice conoscere le coordinate del robot (coordinate intese come posizione del robot all'interno della mappa).

Pianificare i percorsi

Data la mappa rimane il problema di dover pianificare i percorsi che deve eseguire il waiter all'interno di essa.

Sono possibili, in linea di massima, due strade principali:

- Pianificare a priori tutti i percorsi possibili da/a tutti i punti di interesse nella sala.
- Creare un sistema in grado di calcolare i percorsi in modo dinamico

Sebbene la prima soluzione risulti essere più veloce da implementare, essa non è scalabile, e una piccola modifica della posizione di un punto di interesse (come può essere benissimo un tavolo), richiede di ricalcolare a mano tutti i percorsi che coinvolgono tale punto.

Risulta essere più opportuno quindi, fare uso di un sistema che, conoscendo il punto di partenza e il punto di arrivo, calcoli il percorso migliore fra quei due punti, cioè un **planner**.

Per implementare tale sistema si fa uso di conoscenze in intelligenza artificiale, in particolare si fa uso di un *sistema* fornito dalla nostra software house.

Tale sistema è in grado di calcolare il percorso migliore utilizzando l'algoritmo A* partendo dalla conoscenza della posizione del robot nella sala, della mappa della sala e del punto di arrivo.

In questa maniera si ha subito un sistema scalabile e generale, cioè se cambia la sala è sufficiente cambiare la mappa della sala, dato che i percorsi vengono generati dinamicamente.

Walker

Ai livelli appena citati, si aggiunge il livello dal walker da noi introdotto.

La necessità di tale modello è sorta analizzando la problematica dell'indipendenza tra la posizione dei punti di interesse all'interno della sala da tea e le azioni da attuare nei suddetti punti di interesse.

Introdurre un layer, ovvero il **walker**, tra il **waiter** ed il **waiterwalker**, permette di **modificare senza nessuna ripercussione sul sistema**:

- Le posizioni dei punti di interesse all'interno della stanza di tea, modificandone eventualmente anche il numero.
- Ma anche il modo in cui tali informazioni sono memorizzate, aspetto molto importante per un sistema robusto.

Nello specifico il walker può ricevere tali messaggi di alto livello dal waiter:

```
Request moveToEntrance : moveToEntrance(ARG)
Request moveToTable : moveToTable(TABLE)
```

```
Request moveToBarman : moveToBarman(ARG)
Request moveToExit : moveToExit(ARG)
Request moveToHome : moveToHome(ARG)
```

I quali impediscono al waiter di conoscere dettagli implementativi specifici del movimento, ma soltanto di richiedere il movimento stesso.

Infine, il waiter si occupa di recuperare le informazioni relative alla coordinate dei punti di interesse e richiedere l'effettivo movimento **waiterwalker** tramite i messaggi seguenti:

```
Request movetoCell : movetoCell(X,Y)
Reply moveOk : moveOk(ARG)
```

PROJECT

Modello

Si è deciso di rappresentare il modello di riferimento utilizzando il modello qak (per approfondimenti è possibile consultare il materiale disponibile presso (https://github.com/anatali/iss2020LabBo/tree/master/it.unibo.qakactor/userDocs), poichè abbina alla velocità di prototipazione, la possibilità di fruire di una visione chiara e sintetica del sistema in analisi.

Gli attori qak individuati sono: **waiter, smartbell, barman** (e **client** come entità estera per simulare il comportamento di un client all'interno del sistema).

Il modello di dominio qak è consultabile presso:
https://github.com/virtualms/IssProject/tree/master/sprint1

TEST PLANS

I test plan rimarcano quanto introdotto nella fase di analisi dei requisiti. Si controlli quindi che il sistema, ad un certo stimolo di un client, risponda nella maniera aspettata.
I test plan sono disponibili presso il link:
https://github.com/virtualms/IssProject/tree/master/sprint1/tests