PIAZZA\_OVERVIEW INIZIALE

…Nelle parole in ROSSO DEVO AGGIUNGERE i link…

Lo scopo di questo file di testo è quello di iniziare ad affrontare il progetto finale del corso, nel tentativo di impostare un piano di lavoro basato su un approccio di sviluppo incrementale ( o anche detto a spirale) ispirato alla metodologia SCRUM.

Dopo una prima e attenta lettura dei requisiti riporto qui di seguito una primissima fase di Analisi dei Requisiti (con TestPlans), e di Analisi del Problema tentando di affrontare il problema nella sua interezza, seppur al momento partendo da un alto livello di astrazione.

Da questa overview iniziale potrò ricavare una visione panoramica dell’intero problema e questa mi servirà ad impostare un workplan basato su una divisione del lavoro in Sprint.

Per aiutarmi a definire lo Sprint Planning verranno fatte una serie di assunzioni, partendo da alcune più stringenti e coprenti quindi pochi requisiti, e rilassandole man mano, arrivando a coprire progressivamente tutti i requisiti del problema. Ogni Sprint partirà avendo alla base determinate assunzioni. Scelgo di andare dal Semplice 🡪 difficile (Spiegare)

NOTA: Questa overview verrà fatta inizialmente in linguaggio naturale e con l’ausilio di alcune tabelle e/o immagini. Sono consapevole però di quanto sia ambiguo il linguaggio naturale e di quanto sia difficile, se non addirittura impossibile in certe situazioni, tentare di formalizzare un requisito con il linguaggio naturale.

È per questo che mi impegnerò a produrre un modello formale ed eseguibile dei requisiti e (1 del problema oppure uno unico??), considerandolo come frutto di questa overview.

Sarà utile tenerlo presente nell’affrontare i vari Sprint.

ANALISI DEI REQUISITI:

L’obiettivo del progetto è quello di realizzare un prodotto software in grado di il comportamento di un ddr cameriere, detto **waiter**, che si troverà a dover lavorare in una Tea-Room (vedi Figura n.1) e a dover regolare gli accessi dei clienti al tempo del Covid-19.

1. Vocabolario dei termini

|  |  |  |
| --- | --- | --- |
| **Termine** | **Significato inteso dal committente** | **Altre note** |
| waiter | È un differential drive robot che ha il ruolo di cameriere della tea-room. |  |
| tearoom | È una stanza dal perimetro rettangolare e le cui pareti sono libere da ostacoli.  La tea-room deve essere safe, ossia popolata solo da clienti con T°<37.5, contenente dei tavoli opportunamente distanziati e i quali vengono sanificati dopo ogni consumazione. | Vedi Figura n.1 per conoscere la disposizione della tea-room. |
| home | Posizione della tea-room da cui il robot inizia a lavorare e in cui ritorna quando non ha nulla da fare. |  |
| entrancedoor | Posizione della tea-room da cui entrano i clienti. |  |
| exitdoor | Posizione della tea-room da cui escono i clienti. |  |
| teatable1 | Tavolo n.1 nella tea-room in cui un cliente può sedersi e consumare il proprio tea. | È un ostacolo per il waiter. |
| teatable2 | Tavolo n.2 nella tea-room in cui un cliente può sedersi e consumare il proprio tea. | È un ostacolo per il waiter. |
| barman | È l’entità addetta a ricevere le ordinazioni di tea provenienti dal waiter e a realizzarle. |  |
| servicedesk | Posizione della tea-room in cui si trova il barman. |  |
| smartbell | È un campanello situato in prossimità della entrancedoor e tramite il quale i clienti possono notificare al waiter il loro interesse ad entrare nella tea-room.  È in grado di misurare la temperatura corporea del cliente che desidera entrare e assegna al cliente un clientidentifier. |  |
|  |  |  |
| hall | Zona adiacente alla tea-room nella quale i clienti possono transitare o per chiedere di entrare nella tea-room o per defluire dopo esserne usciti. |  |
| presencedetector | È un dispositivo (un sonar) che rileva la presenza di una persona (o di qualche entità) nella hall. |  |
| maxstaytime | Tempo massimo in cui un cliente può consumare il tea se non c’è nessun altro tavolo tableclean. |  |
| maxwaitingtime | Tempo massimo che un cliente deve aspettare prima di entrare nella tea-room con tutti i teatable occupati. | Una stima pessimistica del maximum waiting time può essere fatta ponendolo uguale a maxstaytime (trascurando il tempo del task convoy) ??? |

Gli elementi appartenenti alla tea-room sono…

Le entità individuabili dai requisiti sono…

-waiter

-barman

-smartbell

-cliente?!

-manager?!

I messaggi che si scambiano sono:

I task del waiter sono…

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Descrizione** | **Entità coinvolte** | **Livello di priorità ?? (oppure basta Interrompibile Si/No)** |
| accept | Se c’è almeno un teatable nello stato tableclean il waiter risponde in modo affermativo alla richiesta di un cliente di entrare. | waiter 🡪 cliente(o smartbell?) | No |
| inform | Se non c’è nessun teatable nello stato tableclean il waiter informa il cliente del maximum waiting time. | waiter 🡪 cliente | No |
| reach | Il waiter raggiunge la entrance door e accompagna il cliente al tavolo. | waiter-cliente | No |
| take | Il waiter prende l’ordine del cliente e lo trasmette al barman. | cliente 🡪 waiter 🡪 barman | No |
| serve | Quando l’ordine del clienteID è pronto il barman lo comunica allo waiter, che lo prende e lo porta al tavolo opportunuo. | barman 🡪 waiter 🡪 cliente | No |
| collect | Quando il cliente ha finito di consumare, o è passato maxstaytime il waiter si reca presso il tavolo cliente per farlo pagare. | waiter- cliente | No |
| convoy | Il waiter accompagna il cliente alla exitdoor. | waiter-cliente | No |
| clean | Il waiter pulisce il tavolo. | waiter | Si |
| rest | Il waiter si reca alla home e vi resta fintanto che non ha nulla da fare. | waiter | Si |

Abozzo di TestPlan funzionali significativi:

Potremmo pensare ad un testPlan funzionale per ogni task

…

ANALISI DEL PROBLEMA:

-maxstaytime: Inizia ad essere calcolato da quando gli viene servito il tea!? No direi da quando si siede al tavolo.

-Per evitare fraintendimenti e situazioni scorrette è opportuno che il tavolo passi allo stato ‘busy’ sin dal task accept. Infatti, non appena il waiter accetta una richiesta di ingresso di un cliente, anche se fisicamente il tavolo non è ancora occupato, logicamente lo è già.

-Il waiter dovrà conoscere la pianta della tea-room per essere in grado di muoversi autonomamente al suo interno. 🡪 RoomMap

-Il waiter deve sapere in ogni momento la situazione della stanza, quindi stato dei tavoli… e poi? Ecc 🡪knowledge base e uso di Prolog.

-Discard Message Off perché…

-Note sulla sintassi e forma dei messaggi….

-Il problema richiede di tenere traccia dello stato logico in cui si trova il waiter. Ho realizzato questo diagramma per visualizzare graficamente i possibili stati in cui può trovarsi il waiter e le transizioni tra questi. Al momento possiamo pensare a 3 stati logici principali. Se servirà in futuro potremmo aggiungerne altri.

-rest 🡪 serving client (ClientID) 🡪 cleaning\_table (N)

-Il problema richiede di tenere traccia dello stato dei teatable. Ho realizzato questo diagramma per visualizzare graficamente i possibili stati in cui può trovarsi un teatable e le transizioni tra questi:

*Ingresso di un cliente che occupa il tavolo*

*Clean task*

*del waiter che sanifica il tavolo*

**busy**

*Uscita del cliente che libera il tavolo*

**tableclean**

*(free and clean) clean*

**dirty**

*(free but dirty)*

*“..the waiter should reduce as much as possible the waiting time of the requests coming from each client.”*

-Il problema richiede che lo waiter sia in grado di ridurre al minimo il tempo di attesa di una richiesta proveniente da un cliente. Questo significa che lo waiter dovrà essere in grado di fare interleaving dei suoi task, ossia dovrà essere in grado di valutare autonomamente se, sotto determinate condizioni, sia il caso di interrompere l’attività che sta eseguendo per dedicarsi eventualmente ad un’altra. Dovrà poi essere in grado di riprendere l’attività interrotta dal punto in cui era rimasto e di portarla a termine.

Possiamo pensare alla possibilità di assegnare un livello di priorità ai vari task del waiter a seconda dell’importanza che possono avere nel ridurre l’attesa di un cliente che vuole entrare.?? Oppure basta interrompibile si/no?.

-Lista di assunzioni:

1. Abbiamo un solo cliente da servire (e il waiter conosce già la mappa della stanza?)  
   Il barman è in grado di lavorare alla preparazione di un ordine alla volta. Ordini ulteriori verranno accodati e iniziati a preparare solo una volta finito di preparare il precedente.
2. Il waiter non conosce la mappa della stanza. Prima di iniziare le attività deve costruirsela.
3. Abbiamo più clienti da gestire.  
   I task del waiter non sono interrompibili.
4. Ottimizziamo lo scheduling dei task del waiter: vogliamo ridurre il tempo di attesa delle richieste provenienti dai clienti.

Abstraction gap:

Leggendo i requisiti è abbastanza evidente come il problema che stiamo affrontando descriva **un sistema distribuito composto da diverse entità**. Ciascuna di queste deve essere in grado di agire autonomamente e deve essere in grado di interagire con le altre entità del sistema. Queste caratteristiche sono molto più vicine alla definizione di Attore che alla definizione di oggetto. Gli attori sono entità dotate di flusso di controllo autonomo e che comunicano tra loro non tramite procedure-call, bensì in modo asincrono, a scambio di messaggi. Si presenta dunque l’opportunità di ridurre l’abstraction gap scegliendo di modellare le entità del nostro sistema come attori.

Per ridurre ulteriormente l’abstraction gap e velocizzare il processo di sviluppo si segnala l’opportunità di utilizzare i QActor sfruttando il Domain Specific Language (DSL) QaKActor. Si tratta di un DSL interno che è stato sviluppato nel corso delle lezioni come infrastruttura sopra al linguaggio Kotlin.

Grazie a questo DSL è possibile scrivere dei modelli dell’architettura di un sistema distribuito composto da attori focalizzandoci solo sulla logica applicativa del problema, sul comportamento dei suoi componenti e sulla loro interazione e astraendo dai dettagli implementativi.

Abbiamo così la possibilità di esprimere in modo conciso e formale (eseguibile da un calcolatore) dei modelli dell’architettura del sistema sin dalle prime fasi del processo di sviluppo, dettaglio importantissimo per instaurare un rapporto più produttivo con il committente.

Sarà compito del progettista poi fare zooming sui componenti del modello, concentrandosi sui dettagli implementativi.