## DOMANDE

25/06/2020

* Con workplan intende Product Backog+Sprint planning? *Si, ok.*
* **Conteggio del maxstaytime**: Inizia ad essere calcolato da quando gli viene servito il tea!? Non sarebbe meglio da quando il cliente si siede al tavolo e si prende tempo per ordinare?

*Non deve essere imputato al cliente il tempo ad esempio dovuto alla preparazione del tea.*

*Il tempo per ordinare e per consumare invece sì.*

*Se il cliente passa un’ora a pensare cosa prendere, il waiter deve accompagnarlo all’uscita ma logicamente senza farlo pagare.*

* Nella fase di progetto di uno Sprint possiamo riutilizzare progetti/codice sviluppato a lezione quali basic robot, robot planner ecc… avendo fatto un’analisi opportuna in precedenza?

*Si, una volta impostata l’analisi possiamo usare tutto il codice/progetti che vogliamo.*

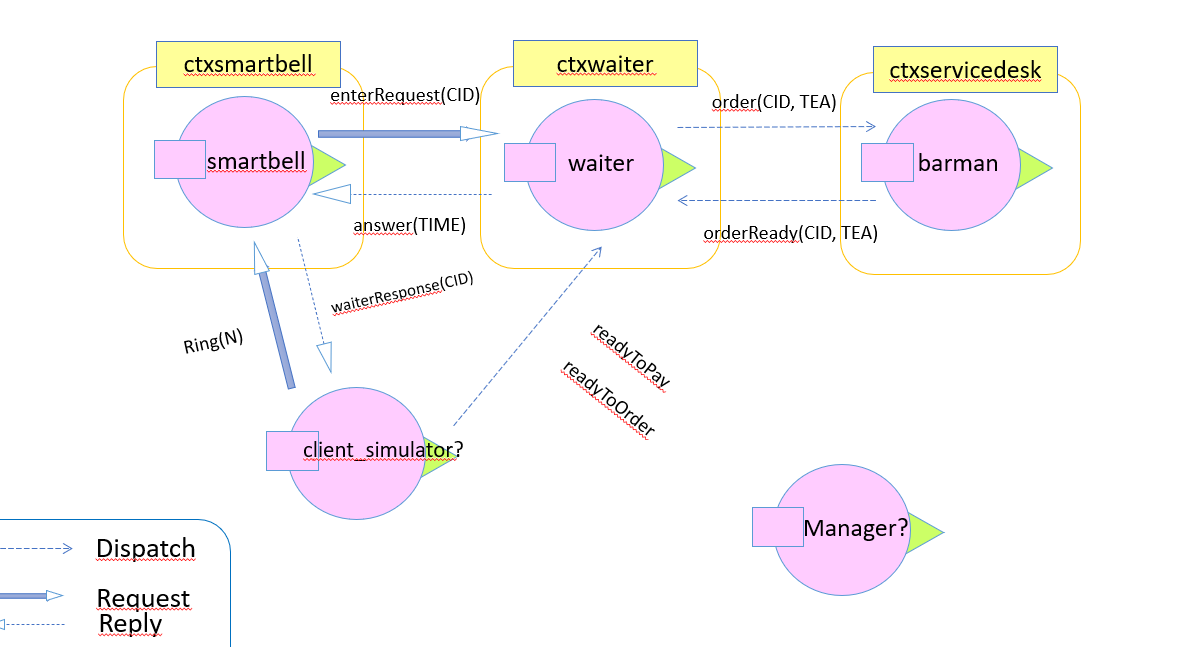
* Parlare di contesti già nell’architettura logica dell’overview iniziale è precoce? Cliente e manager possono avere i loro contesti? …Il manager è inteso come colui che raccoglie le informazioni sulla current state della tearoom. Per ridurre il traffico è conveniente deployarlo nello stesso contesto del ctxwaiter che è quello dove risiedono più informazioni? *Direi di si…*
* *Per rendere veritiera la simulazione del cliente occorre* ***sincronizzare il suo comportamento con quello del waiter.*** *Me ne occuperò in uno dei futuri Sprint.*

In fase di Analisi del Problema devo fermarmi qui o posso anche dire questo che segue?

***readyToOrder e readyToPay come Request:***

*Si è pensato all’opportunità di modellare i messaggi readyToOrder e readyToPay, inviati dal cliente al waiter, come messaggi Request. Infatti, dal momento che l’ordinazione e il pagamento devono avvenire in presenza del waiter al tavolo, per il cliente può risultare utile ricevere una risposta alla sua richiesta che dica “Eccomi, sono qui al tavolo a tua disposizione”. Nel caso del pagamento il payload messaggio di risposta può essere utilizzato anche per specificare il conto da pagare.*

* Cliente e manager devono vanno rappresentati nell’architettura logica?

Quindi questa interazione cerchiata in blu serve o no? Perché i requisiti dicono hitting the smartbell…quindi nella realtà non servirebbero questi 2 messaggi…

Un’alternativa per simulare il cliente potrebbe essere quella di usare degli automessaggi che si manda ad esempio il waiter… Ok forse con un solo cliente…ma per simulare tanti clienti?

*PROBLEMA: SIMULAZIONE CLIENTE E SINCRONIZZAZIONE DELLA SUA INTERAZIONE CON I TASK DEL WAITER.*

*Inizialmente avevo pensato ad un modello che simulasse stato per stato tutto il comportamento del cliente e che scambiasse messaggi con il waiter in modo da potersi sincronizzare e fare le richieste giuste al momento giusto (es: dire che è pronto per ordinare quando è già seduto al tavolo e non quando ci si sta ancora recando). Questa soluzione poi ha il limite che non può simulare più di un cliente?!*

*Una soluzione che prevede un modello di cliente più semplice, che può essere adatta sia a simulare un solo cliente che molti e che risolverebbe anche il problema di sincronizzazione con i task del waiter può essere la seguente: ?!*

*Un attore, chiamato simulator, che non fa altro che stare in ascolto di messaggi provenienti dal (waiter?) dove viene detto:*

*-table\_reached(CID) 🡪 quando lo riceve, dopo tot manda un messaggio readyToOrder(CID)*

*-tea\_served(CID) 🡪 quando lo riceve, dopo tot manda un messaggio readyToPay(CID), solo se non è già stato cacciato per aver superato maxstaytime (come controllarlo?)*

*L’interazione con la smartbell verrebbe poi totalmente rimossa e sostituita con degli auto-msg ring che si manda da sola la smartbell.*

02/07/2020

* La stessa base di conoscenza Prolog può essere condivisa tra più attori? Se per dire uno solo di questi vi scrive e gli altri vi accedono solo in lettura (dovranno poi essere attori facenti parte dello stesso Context per potervi accedere).

*No, meglio rimanere svincolati da possibili problemi e farli comunicare sempre tramite scambio di messaggi. L’attore che vi accede in lettura dovrebbe fare delle consult continue per avere la kb aggiornata ☹. Poi anche se uno vi accede in lettura potrebbe sorgere il problema che “chi mi assicura che quando poi ho letto e mentre processo quei dati non siano magari già cambiati?” Nessuno. A meno che io non sia l’unico attore che legge e scrive su quella kb: in questo caso posso essere sicuro.*

*Si potrebbe pensare, di modellare ogni entità come proprietaria della relativa kb al proprio stato e inerenti.*

*Poi si può interrogare questa in due modi:*

*-tramite Request (lei fa una solve e poi mi risponde con una) -Reply*

*-Gli attori sono risorse Coap quindi potremmo fargli fare un updateResource ogni volta che cambia la kb. E potremmo mettere un Coap Observer che, ogni volta che cambia la base di conoscenza, notifichi chi di dovere.*

* Dal momento che il waiter deve dedicarsi ai clienti ed essere disponibile a rispondere alle loro richieste, se si dedicasse ad osservare il tempo residuo per ogni cliente ne andrebbe della soddisfazione dei clienti, e nessuna delle due attività verrebbe svolta bene.

Questa problematica richiede la presenza di un’entità dedicata ad osservare la situazione del maxStayTime per ogni cliente in sala. Il comportamento di quest’entità deve essere tuttavia subordinato al waiter in quanto è il waiter a conoscere i momenti esatti in cui un timer va fatto partire, interrotto e fatto riprendere. Sarà un’entità slave. ---ANALISI DEL PROBLEMA

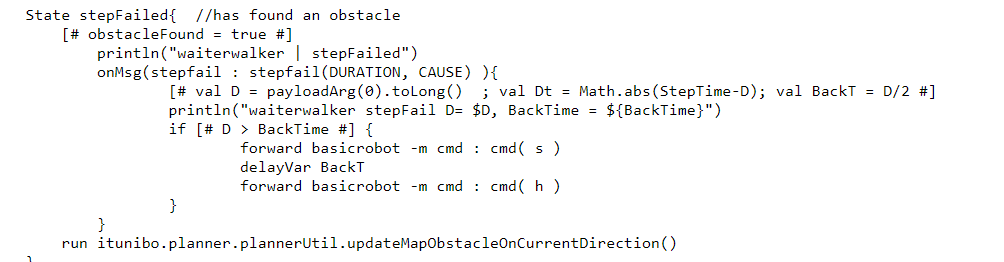
* Osserva solo un cliente oppure tutti?
* Ha la responsabilità di osservare il tempo di permanenza dei clienti…ma di uno solo o di più?
* local-event ? Posso avere tanti sottoattori, non ho problemi di privacy perché un evento locale che rimane dentro al contesto, il waiterlogic non deve conoscere tutti i suoi slave, che magari sono tanti se voglio aggiungere il timer…PROGETTO

ORE 10 SABATO SPRINT\_0-1-2 REVIEW

Si parte dai modelli dei requisiti dei vari gruppi, poi vedere le demo degli Sprint.

Ci vogliono i modelli dei requisiti e di Analisi Problema con i Workspace

Poi dare motivazioni di come ho articolato gli Sprint e le relative assunzioni.

Nel waiterwlaker.qak qual è la logica dietro all’if, ala backT ecc?

GIOCO DA GIOCARE: Come Analista identificare dei sottoproblemi che magari in uno sprint successivo si possono tradurre in veri e propri sottosistemi nati con lo scopo di risolverli.

Essendo in un approccio a spirale, tutto quello che abbiamo fatto in prima battuta è impossibile che sia un’analisi perfettamente corretta. Quindi ad ogni Sprint dobbiamo rileggere l’overview iniziale e magari riprendere una problematica che avevamo suscitato e dire:  
“Con riferimento alla problematica “nome della problematica” evidenziata nell’overview iniziale… la riprendiamo e magari ci siamo accorti che… oppure si amplia perché non avevamo considerato questo e quell’aspetto ecc…”