## DOMANDE

* 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.*

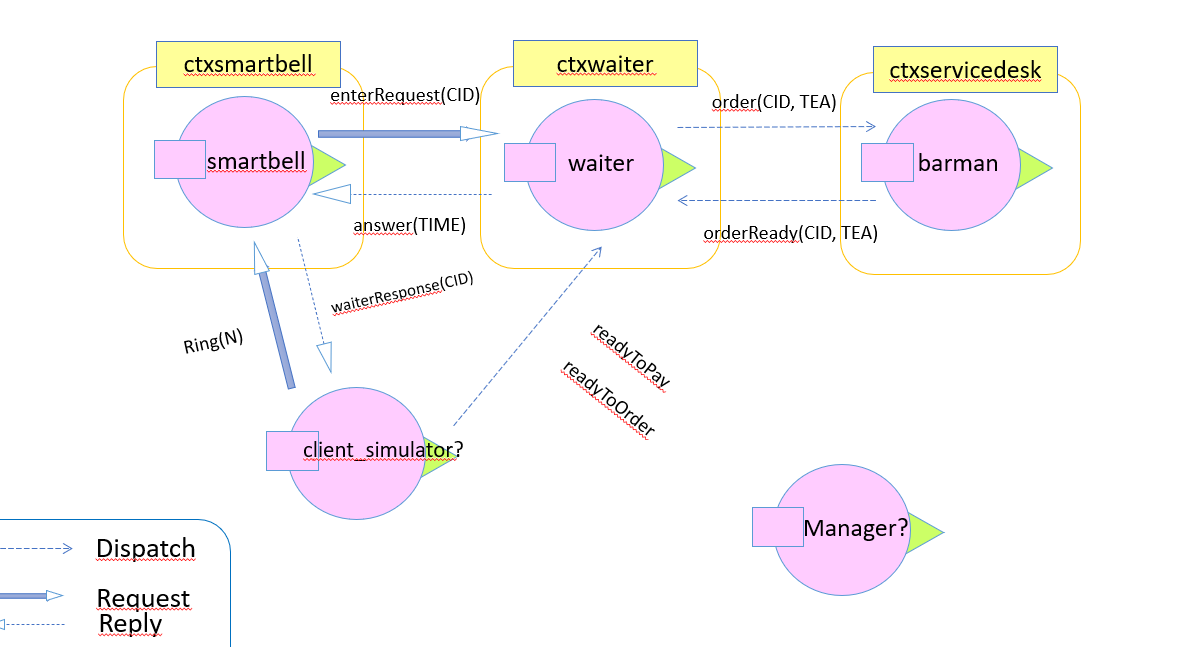
* Parlare di contesti già nell’architettura logica dell’overview iniziale è precoce? Cliente e manager possono avere i loro contesti? …Il manager è inteso come colui che raccoglie le informazioni sulla current state della tearoom. Per ridurre il traffico è conveniente deployarlo nello stesso contesto 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?!*

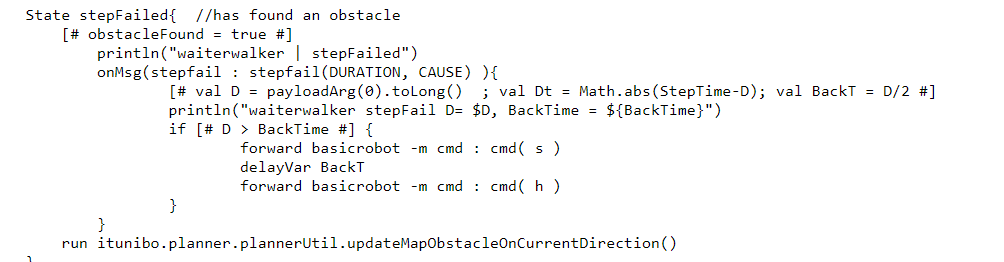
*Una soluzione che prevede un modello di cliente più semplice, che può essere adatta sia a simulare un solo cliente che molti e che risolverebbe anche il problema di sincronizzazione con i task del waiter può essere la seguente: ?!*

*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.*

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…”