# **ANALISI DEI REQUISITI**

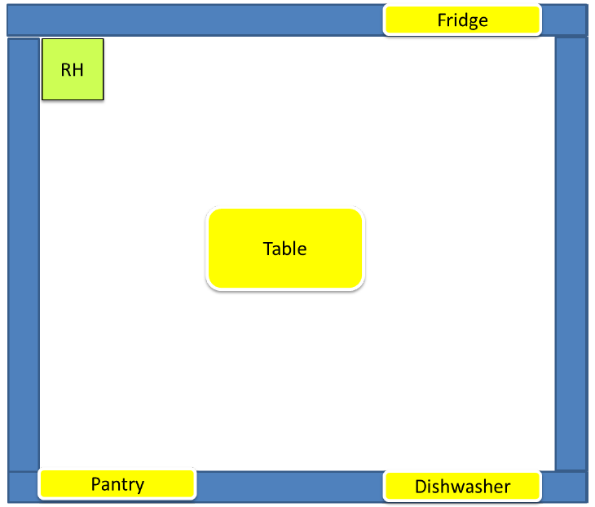
Il maître è dentro la stanza e può dare al robot **tre comandi**: uno di questi è **PREPARE**.

Una volta emesso il comando, il robot:

1. Prima deve andare da RH alla dispensa
2. In seguito dalla dispensa al tavolo per mettere le stoviglie
3. Dal tavolo si reca al frigo
4. Poi dal frigo va al tavolo per mettere il cibo, che ha un **food-code univoco**
5. Infine torna in RH

Con riferimento al documento di analisi dei requisiti del Preliminary Sprint, riportiamo parte dello scenario generale relativo allo sprint corrente:

* **Nello stato iniziale** la lavastoviglie e il tavolo sono vuoti, mentre la dispensa e il frigo contengono un set prefissato di item fornito dal maître. RBR si trova in RH.
* In seguito il maître invia il **comando PREPARE** e attende la fine del task.



52

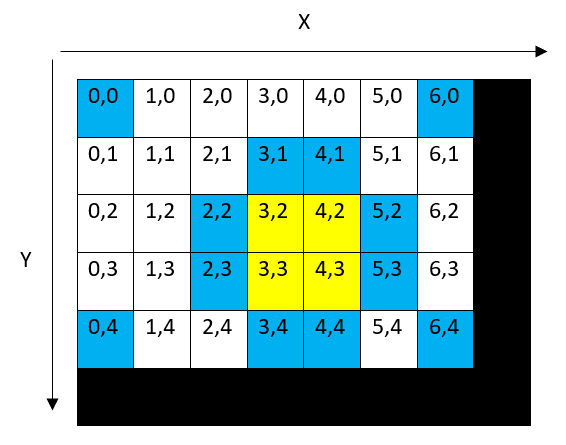
42

1

32

2

# **ANALISI DEL PROBLEMA**

Riprendendo un punto dello sprint 1, ricordiamo che **se la dimensione della stanza e le posizioni degli elettrodomestici sono note**, esse risiedono nella base di conoscenza della stanza. In particolare abbiamo che:

* **Il frigorifero** si trova in (6,0)
* **La lavastoviglie** si trova in (6,4)
* **La dispensa** si trova in (0,4)
* **RH** si trova in (0,0)
* **Il tavolo occupa 4 celle e** si trova in

(3, 2), (4, 2), (3, 3), (4, 3)

e le sue celle di accesso sono

(3, 1), (4, 1), (5, 2), (5, 3),

(3, 4), (4, 4), (2, 2), (2, 3).

Lo sprint 2 nasce perché dall’analisi dei requisiti emerge la necessità di implementare una soluzione per il task relativo al comando “Prepare”. Per soddisfare tale requisito, ci sono diversi problemi da analizzare. A tale scopo si evidenziano le successive note:

1. In seguito ad un chiarimento con il customer**,** sappiamo che **rbr ha capacità infinita.** Ne consegue che con un solo viaggio rbr può portare tutto il necessario da un elettrodomestico/mobile all’altro, senza preoccuparsi della quantità degli item di cui può prendersi carico.
2. Il maitre ha predisposto **un set prefissato di oggetti e cibi** da mettere in fase di esecuzione del task.
3. Nella **dispensa** ci sono **solamente oggetti** e nessun tipo di cibo.
4. Nel **frigorifero** ci sono **solamente cibi** e nessun tipo di oggetto.
5. Con il **comando “prepare**”, rbr:
   1. Porterà gli oggetti dalla dispensa al tavolo
   2. Successivamente porterà i cibi dal frigorifero al tavolo
   3. Al termine, rbr tornerà in RH.

**Ipotesi di Product Backlog:**

* Update KB della dispensa e del tavolo
* Realizzazione logger anche per contenuto delle appliance
* Revisione di metamodello/modello eseguibile
* Realizzazione di un test plan che verifichi gli spostamenti del robot e degli item:
  + Verifica se rbr si sposta da RH a dispensa
  + Verifica se rbr si sposta da tavolo a frigo
  + Verifica se rbr si sposta da frigo a tavolo
  + Verifica se rbr si sposta da tavolo a casa
  + Verifica se le liste di item (oggetti/cibi) si popolano adeguatamente in base agli spostamenti relativi al task
* Implementazione ed esecuzione del test

# **PROGETTO**

Partendo dall’ipotesi di Product Backlog e da quanto scritto in precedenza, ci sono ovviamente diverse possibilità di realizzazione. Noi, in particolare:

* **Per verificare gli spostamenti** di rbr e per avere un tracciamento cronologico di tali spostamenti, abbiamo utilizzato il **logger** realizzato nello sprint precedente.
* **Per gestire il contenuto del frigorifero e della dispensa**, abbiamo aggiunto nella kb della stanza gli item necessari a popolare le liste. Essi, nel file Prolog, sono così formati:
  + **iteminpantry(glasses).** 🡪 per gli item di tipo oggetto, abbiamo ritenuto di dover semplicemente indicare il nome univoco che li caratterizza.
  + **iteminfridge(raspberry, c000, 5).** 🡪 per gli item di tipo cibo, abbiamo rispettato le specifiche e abbiamo inserito un food-code così formato: cxxx in cui x è una cifra numerica. Il numero finale rappresenta invece la quantità disponibile di tale cibo.
* Anche in questo Sprint e nei successivi, continueremo ad utilizzare il **planner** intelligente per raggiungere le nostre celle di accesso.
* Siccome il task relativo al comando “prepare” richiede diversi spostamenti, si presta bene ad essere modellato come una **macchina a stati finiti**. Per tale implementazione è quindi risultata necessaria l’introduzione di un **nuovo attore “prepare**” che esegue proprio il task richiesto dal maître in maniera reattiva.
* **Per verificare invece che le liste degli item (cibi/oggetti)** si popolino adeguatamente in base agli spostamenti relativi al task, realizzeremo un **nuovo logger** parallelo dedicato che scriverà anch’esso in un unico file di log detto “**FileLogPrepare**” (in seguito diventerà più generico chiamandosi “FileLogKb”). La scelta di un unico file di log è la stessa del logger degli spostamenti e cioè mira ad evitareproblemi di concorrenzae vedere cronologicamente quali operazioni vengono fatte sui cibi e sugli oggetti. Tale file ha la seguente **struttura:**
  + Ogni volta che viene aggiunto un oggetto, lo scriviamo nel file.

Esempio: +.OT.knives.

* + Ogni volta che viene rimosso un oggetto, lo scriviamo nel file.

Esempio: -.OR.knives.

* + Ogni volta che viene aggiunto un cibo, lo scriviamo nel file.

Esempio: +.FD.wurstel,c001,19.

* + Ogni volta che viene rimosso un cibo, lo scriviamo nel file.

Esempio: -.FD.wurstel,c001,19.

* + Alla ricezione di ogni comando, rbr scrive il task corrente.
  + Al termine del programma, viene scritto lo stato della stanza.