

Simulazione di un supermercato con Anylogic

Odore Marco

29 ottobre 2017

Docenti: Trubian Marco, Malchiodi Dario
Corso: Simulazione e Teoria delle code

Indice

1	Scopo del progetto	1
2	Agent Based modeling	2
2.1	Gli agenti Customer e genericFood	2
3	Il supermercato e i servizi	3
4	Flow Diagrams	5
4.1	Customer Flow	5
4.2	GenericFood flow	6
5	Statistiche e diagrammi	7
6	Lavori futuri	7

1 Scopo del progetto

L'obiettivo del progetto è quello di simulare, tramite il software Anylogic¹, diverse dinamiche riguardanti un supermercato, come ad esempio il flusso della clientela, la schedulazione del personale e i diversi servizi che possono essere presenti nell'attività.

Il tutto è realizzato tramite la versione *learning edition* del software, che presenta alcune limitazioni, come ad esempio il numero massimo di tipologie definibili per gli agenti e un numero massimo per la loro generazione durante l'esecuzione della simulazione².

¹<https://www.anylogic.com/>

²Durante la simulazione saranno generabili un massimo di 50000 agenti complessivi e in fase di costruzione del modello non è stato possibile definire più di 10 agenti.

2 Agent Based modeling

Data la natura complessa del problema, che possiede moltissime attività parallele e concorrenti da simulare, si è deciso di sfruttare il modello basato su agenti.

Nello specifico sono definiti i seguenti agenti:

- **Customer**: il cliente del supermercato.
- **Worker**: i diversi addetti dei reparti di panetteria, pescheria e macelleria.
- **Warehouseman**: i magazzinieri che si occupano di rifornire gli scaffali.
- **Cashier**: i cassieri per il servizio di pagamento.
- **Cart**: i carrelli utilizzati dai clienti.
- **GenericFood**: la risorsa utilizzata dai magazzinieri per rifornire gli scaffali.
- **AutomaticCashierMachine**: la cassa automatica per il servizio di pagamento.
- **InfoPointHelper**: gli addetti dell'info point.

La maggior parte degli agenti è definita per poterne differenziare l'aspetto all'interno della simulazione, e solo **Customer** e **GenericFood** possiedono un'ulteriore caratterizzazione.

2.1 Gli agenti Customer e genericFood

Il **Customer** possiede diverse variabili e parametri. Nello specifico:

- Variabile **ItemsToBuy**: È un dizionario con coppie Prodotto(String)/Quantità(int), che contiene i prodotti che il cliente vuole comprare e relativa quantità.
- Variabile **Bought**: Un booleano che indica se il cliente ha comprato qualcosa, inizializzato a false.
- Variabile **CounterBuy**: un contatore(int) che indica quanti prodotti il cliente ha nel carrello in quel momento.
- Variabile **NeedsInfo**: Un booleano che indica se il cliente necessita di chiedere informazioni all'infopoint.
- I parametri **needsInfoRate**, **needsMeat**, **needsBread**, **needsFish**, **needsOther**: che rappresentano le diverse probabilità di acquisto (o di richiesta info) che un cliente generico possiede entrando nel supermercato³.

L'agente è inoltre caratterizzato da uno state chart (Figura 1), con tre diversi stati:

³Ad esempio, nella simulazione è stata definita la probabilità che un cliente voglia comprare del pane entrando nel supermercato a 0.7(cioè sette clienti su dieci).

- **InitialState**: lo stato iniziale del cliente entrando nel supermercato.
- **WantsToBuy**: lo stato del cliente quando è in fase di acquisto dei prodotti.
- **WantsToGoAway**: lo stato finale del cliente, quando decide di lasciare il supermercato.

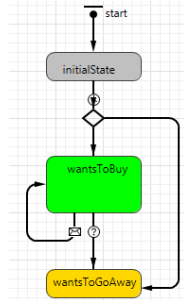


Figura 1: Il diagramma di stato dell'agente Customer.

Quando il cliente entra nello stato **InitialState**, viene eseguito del codice (Figura 2) che permette di inizializzare la variabile **ItemsToBuy** di **Customer**, sfruttando i diversi parametri descritti in precedenza che definiscono le probabilità di acquisto del cliente per i diversi prodotti.

Una volta inizializzata la lista della spesa, se questa è vuota il **Customer** passerà allo stato **WantsToGoAway**, altrimenti entrerà nello stato loop **WantsToBuy**, finché non avrà collezionato tutti gli oggetti, per poi finire anch'esso nello stato finale **WantsToGoAway**.

Per quanto riguarda l'agente **genericFood**, questo possiede due attributi per caratterizzarlo, ovvero:

- **Type**: Di tipo **String**. Specifica la tipologia di prodotto (Es. "meat").
- **Quantity**: Di tipo **Int**. Specifica la quantità di prodotto da acquistare.

3 Il supermercato e i servizi

Il supermercato si estende su una superficie rettangolare, con l'area centrale dedicata agli scaffali per i prodotti generici, la parte destra ai servizi al banco, la parte inferiore ai servizi di cassa, e i magazzini disposti sia sul lato superiore (2) che sul lato sinistro (1). Nella figura 3 è mostrata la planimetria del supermercato e le sue principali aree di interesse.

```

1 shapeBody.setFillColor(black);
2 String []list = {"meat", "bread", "fish", "generic", "generic1", "
   generic2"};
3 ArrayList<String> possibilities = new ArrayList<String>(Arrays.
   asList(list));
4 java.util.Collections.shuffle(possibilities);
5 double res;
6 for(String poss: possibilities){
7     res = uniform(0,1);
8     int quantity = uniform_discr(1,4);
9     if (res<=getParameterFromString(poss))
10    {
11        addObject(poss, quantity);
12    }
13 }
14 res = uniform(0,1);
15 if(res<=needsInfoRate)
16     needsInfo = true;

```

Figura 2: Inizialmente viene eseguito uno shuffle sulla lista di stringhe dei possibili oggetti da acquistare, per differenziare l'ordine di acquisto di ogni cliente, e poi viene simulata un'estrazione da una distribuzione di Bernoulli sfruttando una distribuzione uniforme e il parametro del prodotto di riferimento. Viene inoltre simulata l'estrazione da una distribuzione uniforme discreta (da 1 a 4 oggetti) per la quantità del prodotto da acquistare. Il metodo `getParameterFromString` è una ulteriore funzione custom, definita per il recupero dei valori dei parametri.

Ognuno dei servizi implementati possiede un numero variabile di risorse (rappresentate dagli agenti), che vengono schedate in base all'orario del giorno (ad eccezione delle casse automatiche). Nello specifico abbiamo:

- Servizio al banco per prodotti di panetteria (agenti Worker).
- Servizio al banco per prodotti di macelleria (agenti Worker).
- Servizio al banco per prodotti di pescheria (agenti Worker).
- Servizio di infopoint (agenti InfoPointHelper).
- Servizio di pagamento con cassiere (agenti Cashier).
- Servizio di pagamento con cassa automatica (agenti AutomaticCashier-Machine).

Oltre ai servizi alla clientela sono simulate anche delle attività di rifornimento degli scaffali (agenti WarehouseWorker) per tre diverse tipologie di prodotti (agenti GenericFood), che per comodità, e a causa delle limitazioni della versione learning di anylogic, si differenziano unicamente per il colore (verde, viola, giallo).

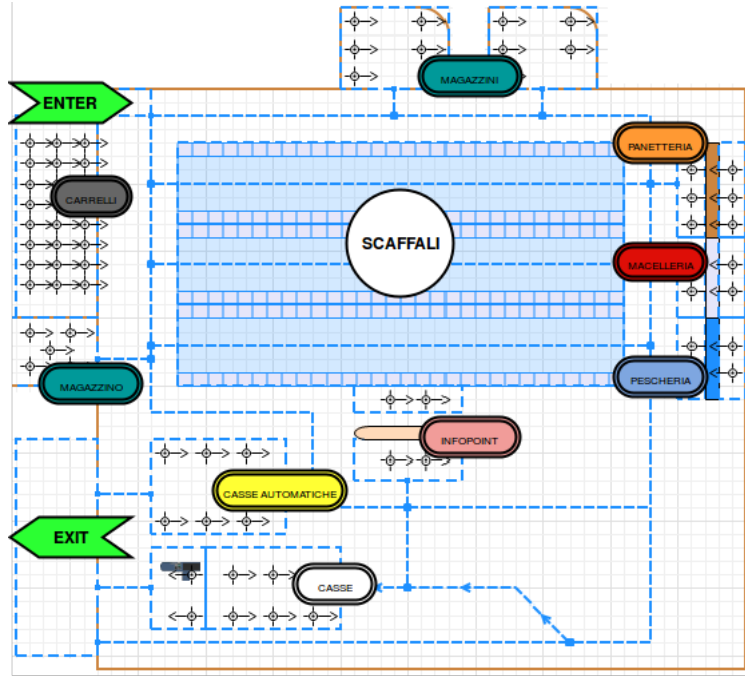


Figura 3: Le principali aree di interesse del supermercato.

4 Flow Diagrams

Per caratterizzare il comportamento dei clienti (agente Customer) e dei prodotti generici da scaffale (genericFood) all'interno del supermercato, sono definiti 4 diagrammi di flusso, di cui 3 dedicati alle tre diverse tipologie di prodotti.

4.1 Customer Flow

I tempi di arrivo (e generazione) dell'agente Customer nel supermercato sono definiti nel flow (Figura 4) all'interno di un blocco sorgente, chiamato **customer-Source**, grazie ad uno scheduling temporale sulle 24 ore. Ogni ora del giorno ha infatti un *rate*⁴ di arrivo specifico. Questo ha permesso di simulare il diverso flusso di clientela che caratterizza la giornata di un supermercato, modulando ad esempio su frequenze più basse gli arrivi notturni, per poi aumentarle durante il giorno (con dei picchi nelle ore di punta).

Il percorso che il Customer seguirà all'interno del flow diagram (e quindi all'interno del supermercato) è definito dal suo stato interno corrente e quindi dal valore delle sue variabili (itemsToBuy, Bought, etc). Il cliente potrà ad esempio decidere se prendere un carrello (se deve comprare più di 10 oggetti), recuperare

⁴Parametro di una distribuzione esponenziale, con media $\frac{1}{rate}$

un prodotto dagli scaffali o dai reparti da banco, pagare alla cassa automatica (se ha meno di 10 oggetti) o a quella con cassieri, se chiedere info all'InfoPoint o , se non deve acquistare nulla, uscire direttamente dal supermercato.

Se il cliente dovrà acquistare dei prodotti, finirà nel loop centrale, che decreterà man mano la sua lista della spesa. Nel momento in cui il cliente farà riferimento ad un servizio (o al recupero di un prodotto da scaffale) manderà poi un messaggio a se stesso contenente la stringa "bought"⁵, che permetterà di uscire dallo stato wantsToBuy definito nello state chart di Customer.

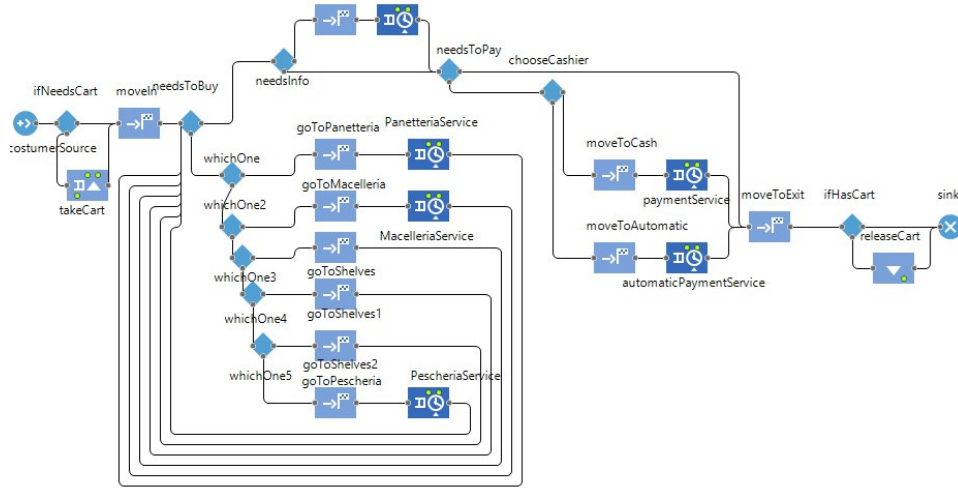


Figura 4: Lo schema di flusso del customer nel supermercato.

4.2 GenericFood Flow

Anche i tempi di arrivo degli agenti genericFood sono regolati da uno scheduling orario (come sempre specificato nel nodo sorgente), che tenta di simulare il rifornimento dei magazzini e degli scaffali. Essendoci tre tipologie per questo agente, sono definiti tre flow corrispondenti (Figura 5).

Il movimento all'interno del supermercato degli agenti genericFood dipende fortemente dalle risorse e dagli agenti che vi interagiscono. Lo spostamento del prodotto sugli scaffali è infatti vincolato alla turnazione dei magazzinieri⁶. Quando

⁵Nello stesso momento verrà eseguita un'istruzione che setterà la variabile Bought del Customer a true, che permetterà di capire che il cliente deve recarsi alle casse per pagare.

⁶Gli agenti/risorse warehouseWorker spostano gli agenti genericFood tramite il nodo RackStore (chiamato putFood nel flow), ma solo se gli scaffali non sono pieni (nodo waitIfFull).

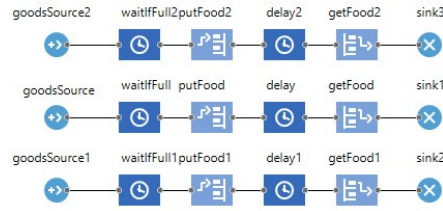


Figura 5: Lo schema di flusso del genericFood nel supermercato.

poi un cliente vuole recuperare uno dei prodotti, lo "sblocca"⁷ dopo la chiamata di una funzione di `stopDelay` (riferita al nodo `Delay` nel flow), per poi poterlo acquistare⁸.

5 Statistiche e diagrammi

Si sono realizzati diversi diagrammi (Figura 6) per ricavare informazioni dalla simulazione, nello specifico:

- Istogramma che mostra le proporzioni totali di utilizzo delle casse automatiche e casse con cassieri. Si basa su delle variabili globali (`totBuy` e `totBuyAut`) che tengono traccia dell'utilizzo delle casse.
- Grafico a torta che mostra le proporzioni totali dei prodotti acquistati nel supermercato (pane, carne, pesce, prodotti generici), che si basa sulla variabile globale `countProducts`.
- Grafico che mostra la lunghezza delle diverse code dei servizi presenti nel supermercato, al tempo corrente.
- Istogrammi per le frequenze di arrivo orario per l'agente `Customer` e per l'agente `genericFood`. Si basano sugli `Histogram Data ArrivalRate` e `ArrivalFoodRate`, che vengono aggiornati nei nodi sorgente, con la funzione `getHourOfDay`.

6 Lavori futuri

Esistono diversi scenari di espansione del progetto:

- L'aggiunta di un servizio rifornimento magazzini con l'arrivo di camion, schedati anch'essi con un rate orario.

⁷Il singolo agente `genericFood` viene rilasciato dallo scaffale (nodo `getFood`) ed eliminato (nodo `Sink` nel flow)

⁸Non è stata gestita la quantità di prodotto richiesta dal singolo `Customer`, nè la situazione in cui lo scaffale non contiene il prodotto richiesto. Di fatto per il `Customer` il prodotto sarà considerato recuperato a prescindere dalla quantità e dalla sua disponibilità.

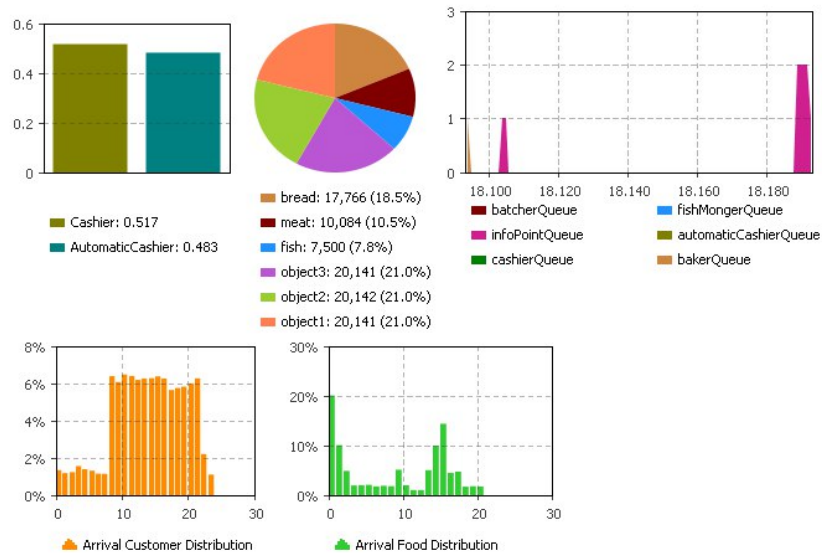


Figura 6: Statistiche sull'esecuzione di una simulazione fino a 50.000 agenti.

- L'aggiunta di diagrammi per i tempi di attesa dei servizi, per la loro ottimizzazione.
- La simulazione dell'impazienza del Customer per un'eventuale attesa troppo lunga.
- La gestione del quantitativo dei prodotti da scaffale e la produzione dei prodotti da banco.
- La simulazione di un incendio/emergenza per la verifica delle vie di fuga.
- L'introduzione di un parcheggio per la clientela (e sua gestione).