TECNICHE DI PROGRAMMAZIONE

14/05/19

Simulazione ad eventi.

Cerchiamo di lavorare con degli strumenti che non ci danno una soluzione propriamente esatta (il problema da affrontare potrebbe essere troppo complesso da risolvere o i vincoli sono troppi e non si riescono a formalizzare tutti) ma che ci permette di simulare degli scenari.

Noi proviamo a capire come si possa scrivere un simulatore in Java per una categoria definita e limitata di sistemi. Abbiamo diversi tipi di simulazioni:

1. Simulazione deterministica o stocastica: se lo scenario è sempre uguale siamo nel primo caso, mentre se dipende da variabili di tipo stocastiche siamo nel secondo caso.
2. Simulazione statica o dinamica: nel caso in cui il sistema dipenda dal tempo siamo nel caso dinamico.
3. Simulazione discreta o continua: se il sistema si evolve continuamente, avremo una simulazione continua mentre se si evolve ad ogni intervallo di tempo ben definito, avremo una simulazione discreta.

Noi, la maggior parte delle volte, affronteremo sistemi discreti in cui potremo identificare un numero finito di istanti di tempo in cui succede qualcosa e in tutti gli altri istanti di tempo non succede nulla. Analizzeremo quindi istanti in cui cambiano le nostre variabili del sistema.

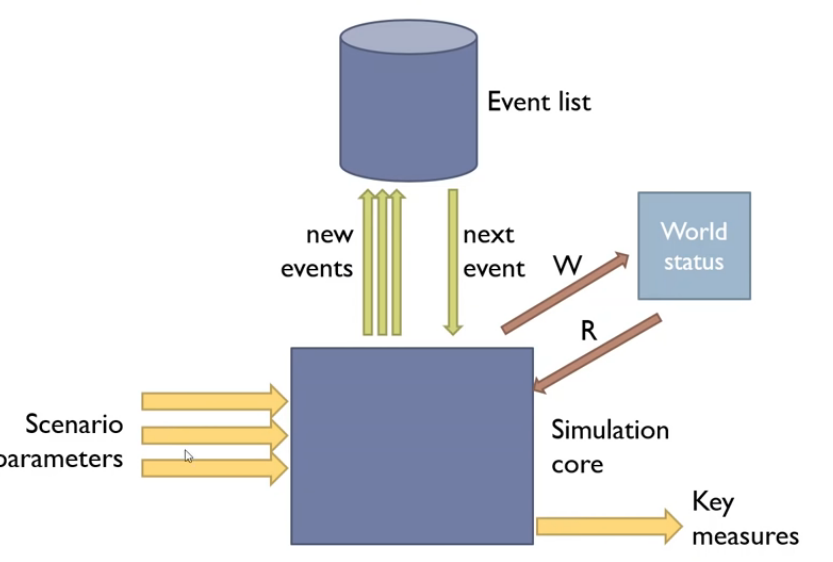
Parliamo quindi di “Simulazione ad Eventi Discreti” (DES), cioè di una simulazione di eventi (dove l’evento è un istante di tempo in cui succede qualcosa) che è Dinamica e Discreta. La DES può essere o Deterministica o Stocastica e gli eventi possono essere correlati tra di loro.

Il modello che analizzeremo noi mantiene una “lista di eventi” in cui, ad ogni step, verrà schedulato un determinato evento mentre un altro verrà processato. Viene rispettata la “priority queue”, infatti viene sempre processato l’evento che presenta priorità maggiore (cioè che viene prima di tutti in termini di tempo). Ogni volta che un evento viene processato, un nuovo evento viene schedulato.

Il simulatore è una macchina che si basa su una lista di eventi e che estrae uno per volta un evento. Dopo averlo elaborato, sulla base di questa elaborazione, genera nuovi eventi (1, 2, tanti, nessuno…) e li butta nella lista. In base alla logica della priorità, il nostro simulatore saprà sempre quale sarà il prossimo evento da processare.

L’elaborazione dell’evento permette al simulatore di aggiornare lo “stato del mondo” (l’insieme delle variabili significative del nostro modello). Il sistema ha bisogno di sapere qual’ era il precedente stato del mondo, quindi deve poter accedere ad esso, ma al tempo stesso può aggiornarlo. Questo vuol dire che il simulatore può sia leggere che scrivere lo “stato del mondo”. Come ultima cosa, il simulatore deve essere modificabile sulla base di una serie di parametri che ci permettono di variare lo scenario e quindi di analizzare diversi casi.

Sulla base di questi elementi, il simulatore deve restituire una serie di valori chiave che ci permettano di decidere quale soluzione sia migliore.



L’evento per noi sarà un Oggetto Java che contiene almeno 3 campi:

1. Tempo: in quale istante di tempo supponiamo che il nostro evento accada? (LocalDateTime o numero intero che si incrementa).
2. Tipo di evento: ci dice cosa dobbiamo fare e rappresenta ciò che accade.
3. Altro: altre informazioni che potrebbero servire in base al tipo di problema o di evento.

Logicamente, il campo più utile è quello del tempo in quanto è quello che conserva l’ordinamento degli eventi.

Di solito, prima di iniziare la simulazione, devo precaricare la coda degli eventi con alcuni eventi futuri.

La simulazione termina o quando ho la coda degli eventi vuota (ho simulato tutto ciò che c’era da simulare) o quando ho raggiunto un tempo limite massimo.

Come ci organizziamo?

Nel nostro programma dovremo creare una classe, separata dalle altre, chiamata “Simulatore”. Sicuramente questa classe dovrà gestire la struttura dati della “coda di eventi” e quella dello “stato del mondo”. Per la prima abbiamo bisogno di una classe “Evento”.

Quindi il Main o il Model crea un Simulatore e gli imposta i parametri. Successivamente dovremo dire al Simulatore di precaricare la coda degli eventi con degli eventi di partenza. Fatto questo il Simulatore è pronto e può partire. Alla fine del “run”, il Main può richiedere i valori finali ottenuti dalla Simulazione.

Internamente, il Simulatore si basa su un Loop in cui, ad ogni iterazione, elabora un evento. Lo “stato del mondo” è un insieme di variabili, liste, grafi che rappresentano la situazione corrente. Lo scopo del Simulatore è quello di far evolvere il Sistema. Lo stato del Mondo potrebbe influenzare il modo in cui vengono processati gli eventi.

Cosa prima cosa quindi chiedo alla Coda degli eventi qual è il prossimo evento. In base al tipo di evento eseguo una serie di operazioni. Mediante un while continuo a chiedere eventi fino a quando non sono terminati o non è terminato il tempo della Simulazione.

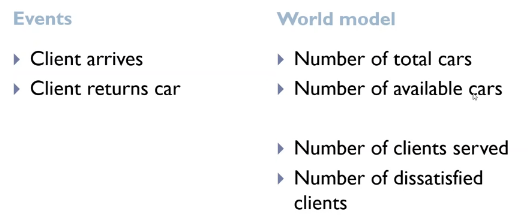
Vediamo ora nella pratica come applicare questo concetto. Immaginiamo di voler aprire un noleggio auto.

Quante auto compro? Più auto ho, più clienti posso soddisfare. Allo stesso tempo però ho fatto un investimento maggiore. Allora qual è la regola che mi collega il numero di clienti che ho con il numero di atuo che devo comprare?

Possiamo provare ad esplorare questo problema supponendo di avere un certo numero di auto e di avere un sistema in cui ogni tot minuti ( per esempio ogni 10 minuti) arriva un nuovo cliente. Quest’ultimo arriva e prende un auto in affito (se è disponibile) per una certa durata. Se invece non ci sono auto, se ne va via e per me è un cliente insoddisfatto.

Ciò che voglio fare è, sulla base dei parametri di simulazione, valutare la mia strategia (quante auto acquistare) sulla base del numero di clienti insoddisfatti.

Il tempo è discreto in quanto succede qualcosa solo quando arriva un cliente per affittare un auto oppure quando un cliente, che aveva affittato un auto precedentemente, la restituisce. Quindi ho solo questi due tipi di eventi.

Lo “stato del mondo”, invece, è caratterizzato dal numero di auto totali, dal numero di auto disponibili, dal numero di clienti serviti e dal numero di clienti insoddisfatti.

Proviamo a scrivere il programma in Java.

Le classi principali che ci servono sono: L’evento, la coda, il Mondo e il Simulatore.

Creiamo come prima cosa un nuovo progretto di tipo JavaFX “NoleggioAuto”.

Immagine che contiene screenshot

Descrizione generata automaticamente

Ricordiamoci di cambiare sempre la struttura del progetto come nell’immagine superiore (premere Next…).  
All’interno di queta project inseriamo in package del Model e all’interno di quest’ultimo creiamo la classe “Simulatore” e la classe “Evento”.

La classe evento deve tener conto del tempo e del tipo di evento. Nel nostro caso non abbiamo degli istanti di tempo predefiniti in cui possono capitare gli eventi, possono capitare lungo tutta la giornata. Possiamo quindi usare una variabile temporale vera e propria tipo “LocalTime”. Per rappresentare il tipo di evento decidiamo di usare una classe “enum” che ha la stessa funzione di una clases Boolean.

Immagine che contiene screenshot

Descrizione generata automaticamente

La variabile “CLIENTE\_ARRIVA” vale 0 mentre “AUTO\_ RESTITUITA” vale 1. A questa classe “enum” agganciamo una variabile privata che contiene appunto uno dei due valori.

Il nostro evento deve essere messo in una coda ordinata per tempo, quindi sicuramente deve implementare la “Comparable” in modo da poter comparare la priorità tra gli eventi.

Oltre a questo devo implementare il metodo “compareTo” che ci dice sulla base di cosa confrontare 2 eventi.

All’interno del Simulatore, invece la variabile da creare è quella sulla Coda degli eventi.

Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

FINE.