

# Esame Software Engineering (AA 2021/22)

01 Giugno 2022

*Enrico Tronci*

*Computer Science Department, Sapienza University of Rome  
Via Salaria 113 - 00198 Roma - Italy*

tronci@di.uniroma1.it

<http://mclab.di.uniroma1.it>

## Esercizio 2 (25 punti)

L'unità di tempo per questo esercizio è il secondo.

Un impianto industriale è monitorato attraverso una rete di sensori (IoT).

Il monitoraggio consiste nel leggere periodicamente i valori dei sensori e nel caso aggiornare i parametri di controllo dell'impianto in modo da mantenere i valori dei sensori entro valori normali.

L'impianto consiste di  $N$  sottosistemi, numerati da 1 ad  $N$ . Per ogni sottosistema ci sono  $Q$  sensori, numerati da 1 a  $Q$  che ne monitorano il comportamento.

I valori dei sensori sono memorizzati in una matrice  $X$  a valori reali di dimensione  $N \times Q$ . Il valore  $X[i, j]$  contiene il valore del sensore  $j$  per il sottosistema  $i$ .

I parametri di controllo dell'impianto sono memorizzati in una matrice  $U$  a valori interi di dimensione  $N \times Q$ . Il valore  $U[i, j]$  contiene la strategia di controllo in uso per la dinamica monitorata dal sensore  $j$  del sottosistema  $i$ .

La dinamica di ciascun sensore è modellata con una DTMC. Sia, per ogni  $t$ ,  $r(t)$  una variabile random reale uniformemente distribuita in  $[0, 1]$ . Ogni  $T$  secondi, per ogni  $i \in \{1, \dots, N\}$ ,  $j \in \{1, \dots, Q\}$ , il valore di  $X[i, j]$  è aggiornato come segue:

$$X[i, j](t+1) = \begin{cases} X[i, j](t) + T0.01(-2 + 2r(t)), & \text{if } (U[i, j] = -1) \\ X[i, j](t) + T0.01r(t)2, & \text{if } (U[i, j] = 1) \\ X[i, j](t) + T0.01(-1 + 2r(t)), & \text{otherwise} \end{cases}$$

dove:

$$X[i, j](0) = \alpha(i, j) \tag{1}$$

$$\alpha(i, j) = Q(i-1) + j \tag{2}$$

Potete assumere  $T = 10$ .

L'ambiente **Env** per il software consiste di un server che, oltre alle matrici di cui sopra, mantiene una variabile  $p$  che definisce il sottosistema sotto osservazione ed una  $v$  che definisce il sensore sotto osservazione.

Il server **Env** riceve comandi nella forma  $(c, d)$ , dove  $c$  e  $d$  sono interi. Se  $c = 0$  allora il server assegna alla variabile  $p$  il valore  $d$ . Se  $c = 1$  allora il server assegna alla variabile  $v$  il valore  $d$ . Se  $c = 2$  allora il server assegna alla variabile

$U[p, v]$  il valore  $d$ , cioè aggiorna la strategia di controllo per il sottosistema  $p$ , relativamente al sensore  $v$ . Se  $c = 3$  allora il server ritorna in output il valore reale  $X[p, v]$ .

Il server **Env** è connesso al client di monitoraggio attraverso due fifo: una dal server al client (che invia al client il valore  $X[p, v]$ ) e l'altra dal client al server (che invia al server la coppia  $(c, d)$ ).

Ogni  $T$  secondi il client legge il valore di sensore per un sottosistema in modo che dopo  $W = N * Q * T$  secondi ha letto i valori di tutti i sensori per tutti i sottosistemi.

Sulla base delle letture di cui sopra, ogni  $W$  secondi il client aggiorna la strategia di controllo in modo da mantenere il valore del sensore  $j$  per il sottosistema  $i$  nell'intervallo  $[\alpha(i, j) - 0.5, \alpha(i, j) + 0.5]$ .

Nello specifico, il client manda al server -1 (cioè richiede  $U[i, j] = -1$ ) se il valore del sensore è maggiore di  $\alpha(i, j) + 0.5$ , manda 1 (cioè richiede  $U[i, j] = 1$ ) se il valore del sensore è minore di  $\alpha(i, j) - 0.5$ , manda 0 (cioè richiede  $U[i, j] = 0$ ) in tutti gli altri casi.

Le richieste e gli aggiornamenti vengono effettuati nel seguente ordine. Prima si considerano i sensori per il sottosistema 1, poi quelli per il sottosistema 2, etc fino al sottosistema  $N$ . I sensori vengono considerati dal più piccolo (indice 1) al più grande (indice  $Q$ )

Questo esercizio si focalizza sulla modellazione del server **Env** di cui sopra. A tal fine si sviluppano i seguenti blocchi.

1. Blocco **Env** nel file **env.mo** che modella l'environment descritto sopra.
2. Blocco **Server2Client** nel file **fifos2c.mo** che modella la fifo dal server al client.
3. Blocco **Client2Server** nel file **fifoc2s.mo** che modella la fifo dal client al server.
4. Blocco **ClientMockup** nel file **mockup.mo** che contiene un mock-up del client descritto sopra. Il **ClientMockup** semplicemente manda richieste a random al server e riceve la risposta dal server. La sua funzione è quella di permettere di testare il server.

## NOTA BENE

1. Tutti i parametri del vostro modello devono essere contenuti nel record **Prm** nel file **parameters.mo**. Oltre a quelli menzionati nel testo dell'esercizio potete aggiugnere dei vostri parametri, ma non dovete in alcun caso rimuovere quelli che ci sono poichè vengono usati per la correzione.
2. Il modello **System** nel file **system.mo** deve essere esteso come serve, ma non devono essere rimosso il contenuto già presente poichè viene usato per la correzione.

3. Potete aggiungere file a vostra discrezione ed estendere a vostra discrezione il contenuto dei file che vi sono forniti.
4. Salvo esplicita istruzione in senso contrario, non potete modificare in alcun modo il contenuto già presente nei file che vi sono forniti. Questi vengono usati per interfacciarsi con gli script di correzione. Una modifica delle interfacce fornite rende impossibile la correzione e quindi l'esercizio riceverà 0 punti.
5. Prima di consegnare accertarsi che il vostro modello compili. I modelli che non compilano ricevono 0 punti.