Esame Software Engineering
(AA 2021/22)

08 Aprile 2022, ore 14.00,

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.

Si vuole realizzare un sistema di telemedicina che tiene sotto osservazione alcuni parametri vitali (ad esempio, pressione, valore glicemico, etc) per i pazienti monitorati.

Il monitoraggio consiste nel leggere periodicamente i valori dei parametri vitali dei pazienti ed aggiornare la terapia nel caso qualche parametro sia fuori dal range normale.

Ci sono N pazienti, numerati da 1 ad N e Q parametri vitali, numerati da 1 a Q,

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

I codici delle terapie sono memorizzati in una matrice U a valori interi di dimensione $N \times Q$. Il valore $U[i,j]$ contiene la terapia in uso per il parametro vitale j per il paziente i.

La dinamica di ciascun parametro vitali 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 $i \in \{1, \ldots N\}$, $j \in \{1, \ldots Q\}$, 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 paziente sotto osservazione ed una v che definisce il parametro vitale 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 terapia per il paziente sotto osservazione p, relativamente al parametro vitale 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 minuto (cioè 60 secondi) il client legge i parametri vitali di tutti i pazienti ed aggiorna la terapia in modo da mantenere il valore del parametro j per il paziente i nell'intervallo $[\alpha(i, j)-0.5, \alpha(i, j)+0.5]$. Nello specifico, il client manda al server -1 se il parametro vitale è maggiore di $\alpha(i, j) + 0.5$, manda 1 se il parametro vitale è minore di $\alpha(i, j) - 0.5$, manda 0 in tutti gli altri casi.

Le richieste e gli aggiornamenti vengono effettuati nel seguente ordine. Prima si considerano i parametri vitali per il paziente 1, poi quelli per il paziente 2, etc fino al paziente N. I parametri vitali 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 sviluppino 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 blocco `Probe` nel file `probe.mo` prende come input tutti gli outputs dei blocchi del vostro modello. Questo file viene usato per la correzione del progetto.
3. 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.

4. Potete aggiungere file a vostra discrezione ed estendere a vostra discrezione il contenuto dei file che vi sono forniti.

5. 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 interfaccie fornite rende impossibile la correzione e quindi l'esercizio riceverà 0 punti.

6. Prima di consegnare accertarsi che il vostro modello compili. I modelli che non compilano ricevono 0 punti.