NOME _____ COGNOME _____ MATRICOLA_____

Ingegneria del Software II

18 Dicembre 2013

Parte teoria, punti 14 - Tempo a disposizione: 1h

Esercizio 1 (punti 7)
Si descriva il concetto di Modularità nella fase di progettazione, e si definiscano le possibili relazioni tra moduli.

Esercizio 2 (punti 7)
Si definiscano e si descrivano alcuni modelli di stima dei costi della produzione software e in particolare i modelli COCOMO e COCOMO II.

NOME _____ COGNOME _____ MATRICOLA_____

Ingegneria del Software II
18 Dicembre 2013
Parte pratica, punti 18 - Tempo a disposizione: 2h

Si svolgano gli esercizi 3 e 4 su un foglio e il 5 su un foglio separato

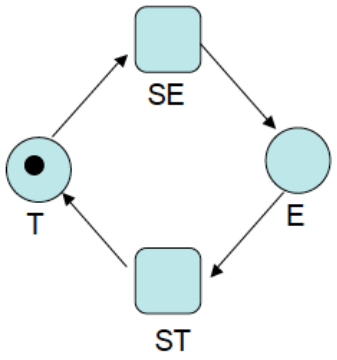
Esercizio 3 (punti 5)
Si modelli con una rete di Petri il famoso problema dei "Dining philosophers": ci sono 4 filosofi, a cena, e 4 forchette. Ogni filosofo pensa, oppure mangia.
Per il modello del singolo filosofo si adotti il seguente:
Due posti:

- T: Thinking
- E: Eating

Due transizioni:

- SE: Start Eating
- ST: Start Thinking

La porzione di PN che modella un filosofo è quindi:



Nella marcatura iniziale tutti i filosofi pensano.
Per mangiare ogni filosofo ha bisogno di due forchette.
Inizialmente tutti i filosofi pensano e nessuna forchetta è impugnata.
Definire il modello complessivo come Rete di Petri.

NOME _____ COGNOME _____ MATRICOLA_____

Esercizio 4 (punti 7)

Un ambulatorio dentistico ha quattro poltrone. Nell'ambulatorio lavorano vari odontoiatri. Ciascuno di loro può lavorare su una sola poltrona. Le poltrone sono assegnate giornalmente, previa prenotazione, a chi ne fa richiesta. Le poltrone sono disponibili dalle 8.00 sino alle 19.00 per periodi di un'ora.

Per poter effettuare una prenotazione è necessario fornire il nome dell'odontoiatria e l'ora. Un odontoiatra non può comparire in più di una prenotazione di poltrone per la stessa ora.

La prenotazione va a buon fine solo se esiste una poltrona libera per l'ora indicata.

Per semplicità si ipotizzi che:

1) Le prenotazioni possano essere effettuate solo per il giorno corrente;
2) L'ora sia individuata da un intero (compreso fra 8 e 18).

Si specifichi in Z l'operazione di prenotazione di una poltrona odontoiatrica.

NOME _____ COGNOME _____ MATRICOLA_____

Esercizio 5 (punti 6)

a) Si esegua l'analisi di flusso, e si trovino le espressioni regolari D-U per le variabili del seguente programma. Cosa suggerisce tale risultato?

b) Si individui, inoltre, un'insieme minimo di casi di test che soddisfi il criterio di copertura dei comandi (motivando l'insieme identificato).

		X	Y	Z
1	#include<stdio.h>			
2	void main (void) {			
3	int X, Y, Z;			
4	scanf("%d", &X);			
5	scanf("%d", &Y);			
6	if (X>=Y)			
7	Z=X-Y ;			
8	else Y= Y- X;			
9	while (Y > 0) do			
10	{ Y= Y-1;			
11	Z= Y – X ; }			
12	if (Z >= 0)			
13	Z= X+Y + Z;			
14	printf("%d\n", Z);			
15	}			

NOME _____ COGNOME _____ MATRICOLA_____

Soluzione

Eserciziom 3

Un filosofo:

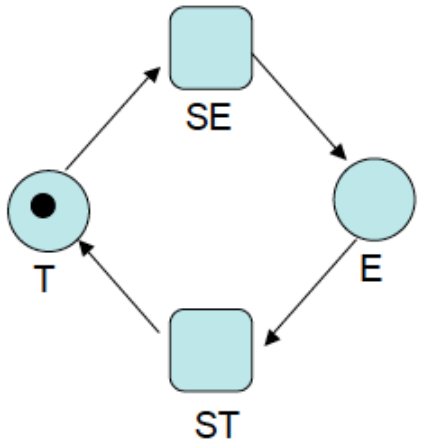
Due posti:

T: Thinking
E: Eating

Due transizioni:

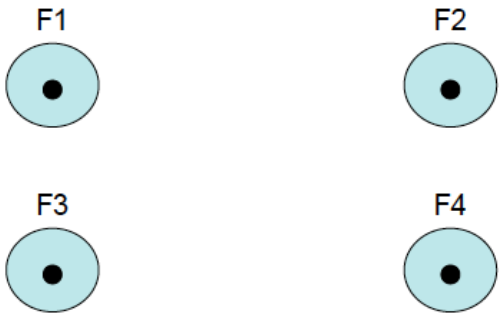
SE: Start Eating
ST: Start Thinking

La porzione di PN che modella un filosofo è quindi:



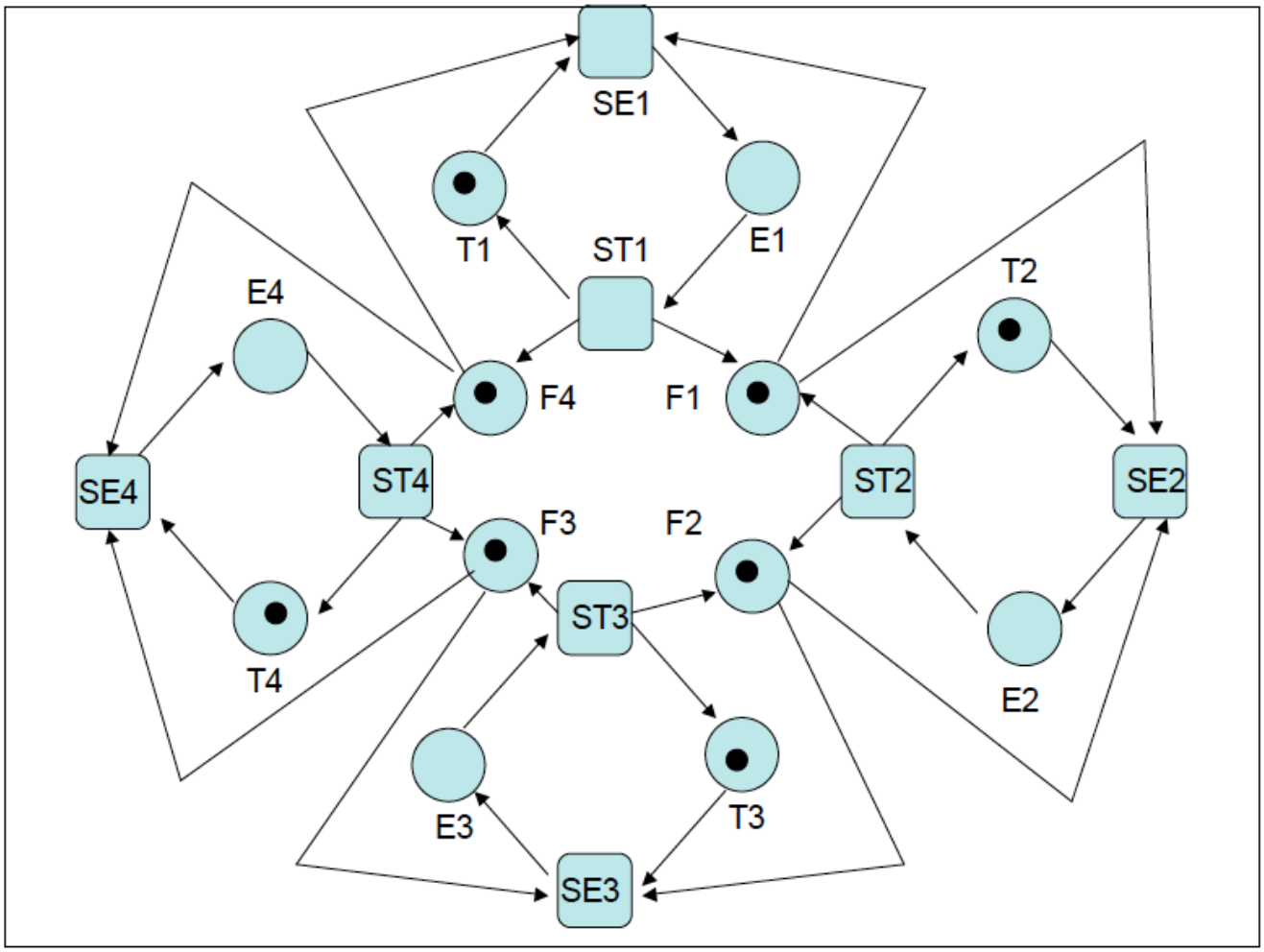
Nella marcatura iniziale tutti i filosofi pensano.

Le risorse sono le 4 forchette, modellabili con 4 posti, F1, F2, F3, F4
(ciascuno marcato nella marcatura iniziale)



La rete complessiva risulta quindi:

NOME _____ COGNOME _____ MATRICOLA_____



NOME _____ COGNOME _____ MATRICOLA_____

Esercizio 4

1) Prenotazione

Tipi definiti dall'utente:

Odontoiatri = insieme degli odontoiatri

Orari= 9…18

Poltrone = 1..4

Variabili che descrivono lo stato del sistema:

1. prenotazioni: funzione parziale dai Orari e Poltrone a Odontoiatri

Ambulatorio

$$\text{prenotazioni}: \text{Orari} \times \text{Poltrone} \pfun \text{Odontoiatri}$$

$$\forall\ o:\text{Odontoiatri}, \text{ora}: \text{Orari} \bullet |\{p:\text{Poltrone} \mid ((\text{ora},p) \mapsto o) \in \text{prenotazioni}\}| \leqslant 1$$

InitAmbulatorio

$$\Delta\text{Ambulatorio}$$

$$\text{prenotazioni}' = \emptyset$$

Success

$$\text{rep}!: \text{Report}$$

$$\text{rep}! = \text{'Okay'}$$

NOME _____ COGNOME _____ MATRICOLA_____

1) Prenotazione

PrenotazioneOK

ΔAmbulatorio
odontoiatra?: Odontoiatri
ora?: Orari
poltrona!: Poltrone

(ora?,poltrona!) $\notin$ dom prenotazioni
$|\{p:Poltrone \mid ((ora?,p) \mapsto odontoiatra?) \in prenotazioni\}| \leqslant 1$
prenotazioni' = prenotazioni $\cup$ {(ora?,poltrona!) $\mapsto$ (odontoiatra?) }
soci' = soci

NessunaPoltronaLibera

ΞAmbulatorio
odontoiatra?: Odontoiatri
ora?: Orari
rep!: Report

$\nexists$ p: Poltrone $\bullet$ (ora?,p) $\notin$ dom prenotazioni
rep! = 'Nessuna poltrona libera'

UnaPoltronaOccupata

ΞAmbulatorio
odontoiatra?: Odontoiatri
ora?: Orari
rep!: Report

$|\{p:Poltrone \mid ((ora?,p) \mapsto odontoiatra?) \in prenotazioni\}| = 1$
rep! = 'L'odontoiatra ha già occupato una poltrona

Prenotazione $\widehat{=}$ PrenotazioneOK $\wedge$ Success

$\vee$

NessunaPoltronaLibera

$\vee$

UnaPoltronaOccupata

NOME _____ COGNOME _____ MATRICOLA_____

Esercizio 5

		X	Y	Z
1	#include<stdio.h>			
2	void main (void) {			
3	int X, Y, Z;	a	a	a
4	scanf("%d", &X);	d		
5	scanf("%d", &Y);		d	
6	if (X>=Y)	u	u	
7	Z=X-Y ;	u	u	d
8	else Y= Y- X;	u	ud	
9	while (Y > 0) do		u	
10	{ Y= Y-1;		ud	
11	Z= Y – X ; }	u	u	d
12	if (Z >= 0)			u
13	Z= X+Y + Z;	u	u	ud
14	printf("%d\n", Z);			u
15	}			

X: adu (u + u) (u)* (u + ε)
Y: adu (u +ud) u (uduu)* (u + ε)
Z: a (d + ε) (d)* u (ud + ε) u

Si nota che per la variabile Z, può esistere un caso di uso senza che sia mai stata definita, ovvero se si esegue il ramo “else” del primo “if” e non si entra nel ciclo “while”.
Inoltre, possono esserci casi di definizioni multiple senza usi intermedi, nelle righe 7 e 11, ma questa anomalia non costituisce necessariamente un errore.

Per eseguire tutti i comandi sarà necessario avere un caso di test con X>=Y in modo da entrare nel ramo “then” del primo “if”, in questo stesso caso potremmo porre Y<0 in modo da non entrare nel ciclo “while” ed eseguire la riga 15 (sicuramente Z>=0 perché alla riga 7 l’abbiamo definito uguale a X-Y). Questo caso di test sarebbe quindi (X=1, Y=-1)
Rimangono da testare la riga 8, che viene eseguita se X<Y, e il ciclo while almeno una volta, che verrà eseguito se Y-X>0, che è la stessa condizione. Un caso di test sarebbe (X=1,Y=2).