CURS 2

Modelarea Fluxurilor de Activitati

Concepte de modelare (1)

- caz / situatie = proces real ce va fi modelat (situatie de afaceri, proces industrial, …)

- Fiecare caz este caracterizat de:
  - un timp de viata limitat
  - starea in care se gaseste, dependenta de evolutia cazului

Concepte de modelare (2)

- Starea in care se gaseste un caz este determinata de 3 elemente:
  - proprietatile / atributele cazului (valorile se pot modifica pe masura ce cazul evolueaza)
  - conditiile ce trebuie indeplinite (pt intrarea sau iesirea din stare)
  - continutul cazului (documente, arhive, baze de date)

Task-ul

- task = unitate logica de lucru, indivizibila, care este executata ca un intreg
- O eroare in executia unui task necesita un rollback pentru reluarea task-ului

- Task-urile pot fi:
  - manuale
  - automate
  - semi-automate

Proces

- proces = procedura asociata unui caz
- Un proces este alcatuit din:
  - task-urile ce trebuie executate
  - conditiile care determina ordinea de executie
  - subprocese
- Ciclul de viata al unui caz este definit de proces

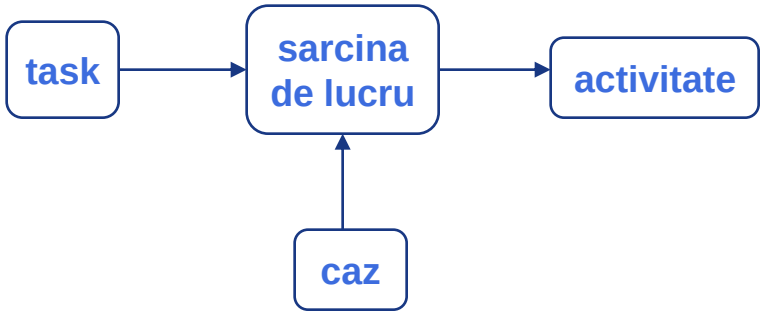
Rutarea

- rutare = constructia ramurilor de executie a unui proces

- Exista 4 tipuri de executie a task-urilor:
  - rutare secventiala
  - rutare paralela (AND-split, AND-join)
  - rutare selectiva (OR-split, OR-join)
  - rutare iterativa

Mecanisme de declansare

- Task-ul este o unitate de lucru generica
- sarcina de lucru = task-ul particular unui proces
- activitate = executia efectiva a unei sarcini de lucru
- O sarcina de lucru intra in executie direct sau prin intermediul unor declansatori:
  - initiativa unei resurse
  - un eveniment extern
  - un semnal de timp



Formalism de modelare

- Evolutia unui sistem poate fi reprezentata in termeni de stari, conditii si evenimente

- Formalismul ales: retele Petri
- Retelele Petri permit reprezentarea grafica a proceselor in vederea modelarii si analizarii lor
- Forteaza o definire precisa
- Baza matematica precisa
- Formalism in intregime definit -> proprietati clare
- Permite utilizarea unor tehnici analitice de evaluare a performantelor

Retele Petri clasice (1)

- O retea Petri este un graf orientat bipartit care poate fi complet definit ca:

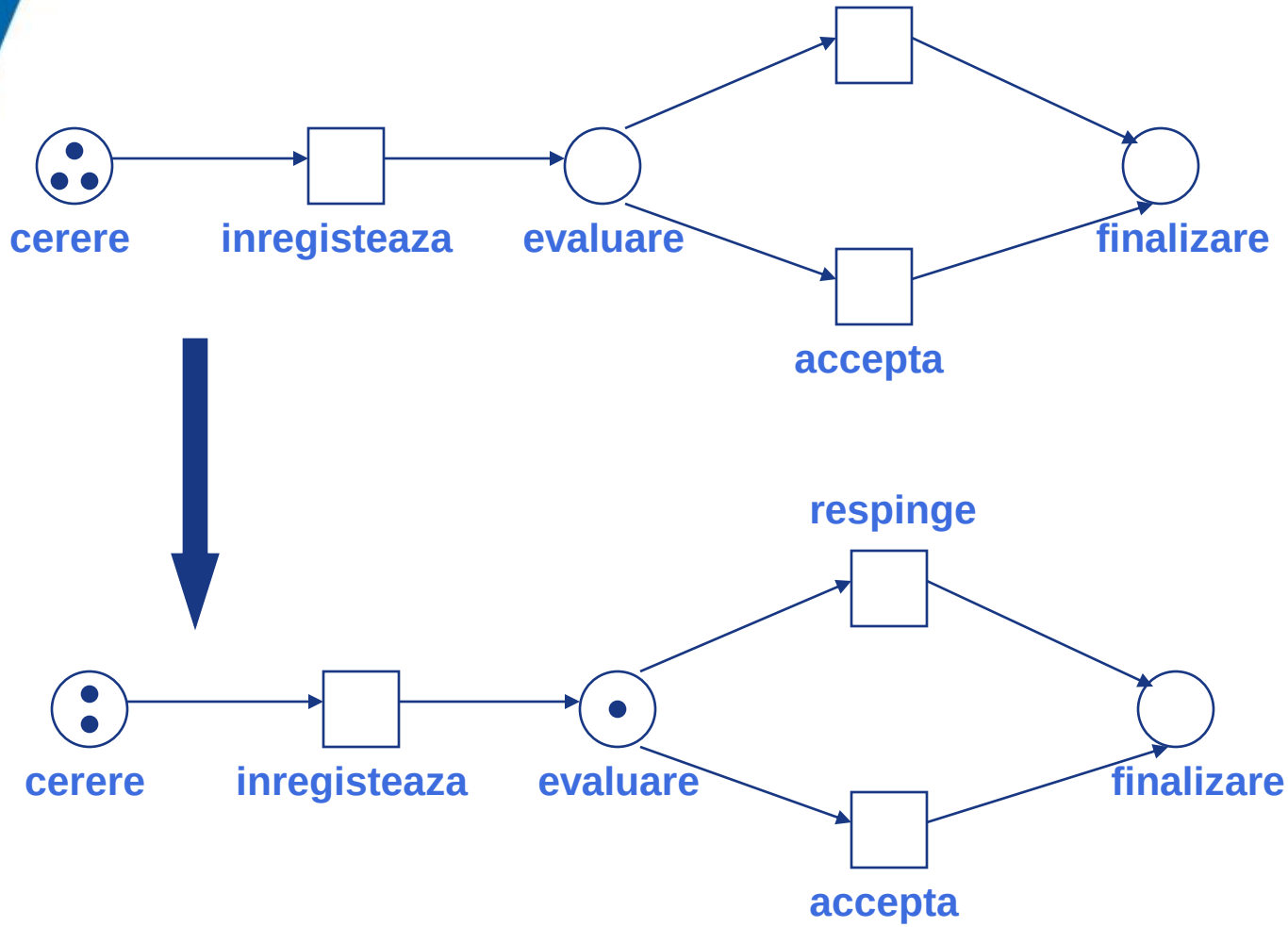
$$RP = (P, T, IN, OUT, M_0)$$

- P = multimea pozitiilor (finita)
- T = multimea tranzitiilor (finita)
- IN, OUT: PxT → {0, 1}
- M_0 = marcaj initial

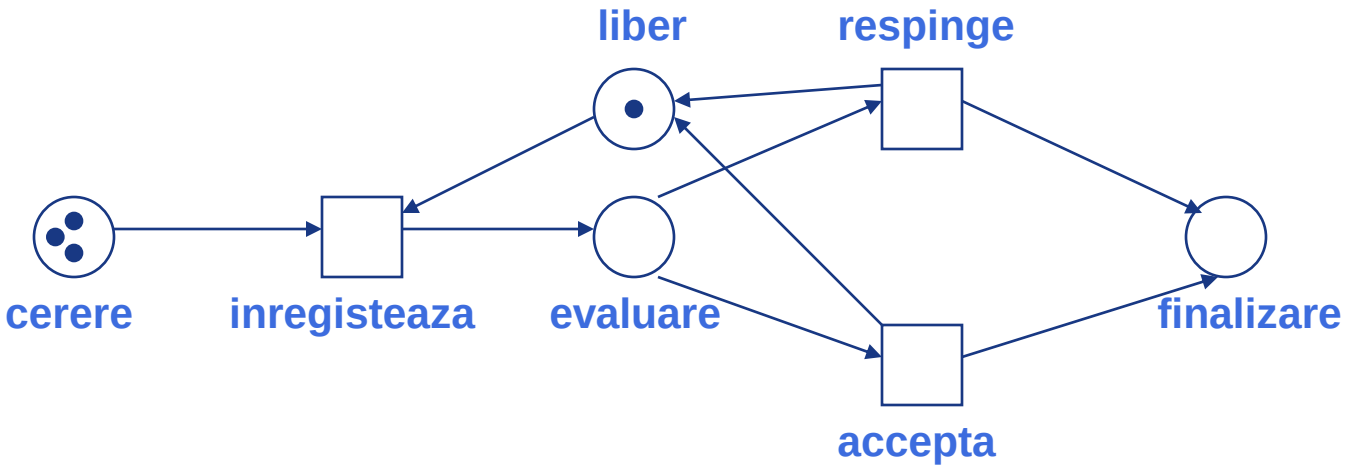
Retele Petri clasice (2)

- pozitiile pot fi folosite pentru a modela:
  - variabile de stare
  - conditii
  - actiuni in desfasurare
  - medii de comunicare, buffere, locatii
- tranzitiile reprezinta:
  - evenimente
  - transformarea unor obiecte
  - transportul unor obiecte

Exemplu (1)


cerere
inregisteaza
evaluare
finalizare
accepta
respinge
cerere
inregisteaza
evaluare
finalizare
accepta

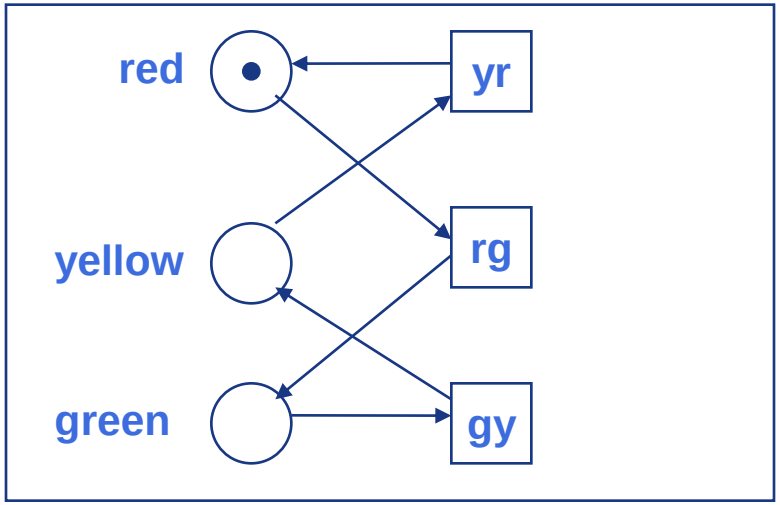
Exemplu (2)



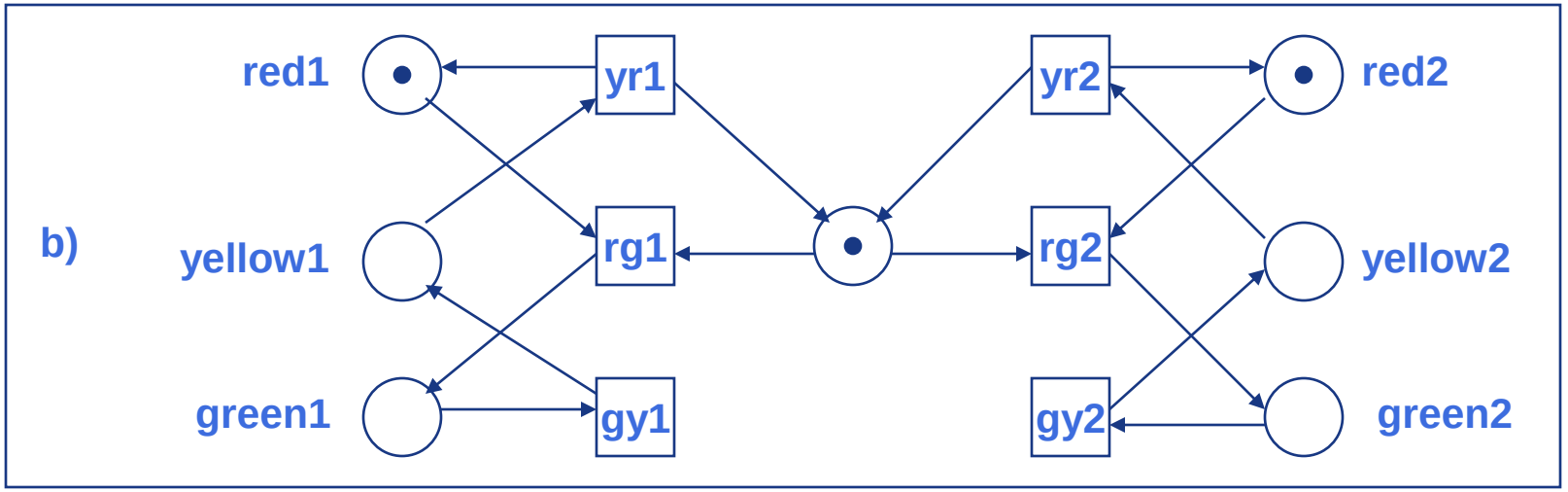
- O tranzitie devine activa daca exista cel putin un jeton de marcaj in fiecare pozitie de intrare pentru acea tranzitie

Sincronizarea semafoarelor

a)



b)



Retele Petri de nivel inalt

Neajunsuri ale retelelor Petri clasice:

- nu se poate face diferentierea intre jetoanele aflate intr-o pozitie
- nu pot fi modelate anumite procese mai complexe
- modelele construite devin prea mari sau inaccesibile

Consecinta: extinderea RP clasice

Extensii ale RP clasice

- Retele Petri colorate
- Retele Petri temporizate
- Retele Petri ierarhizate

Retele Petri colorate (RPC)

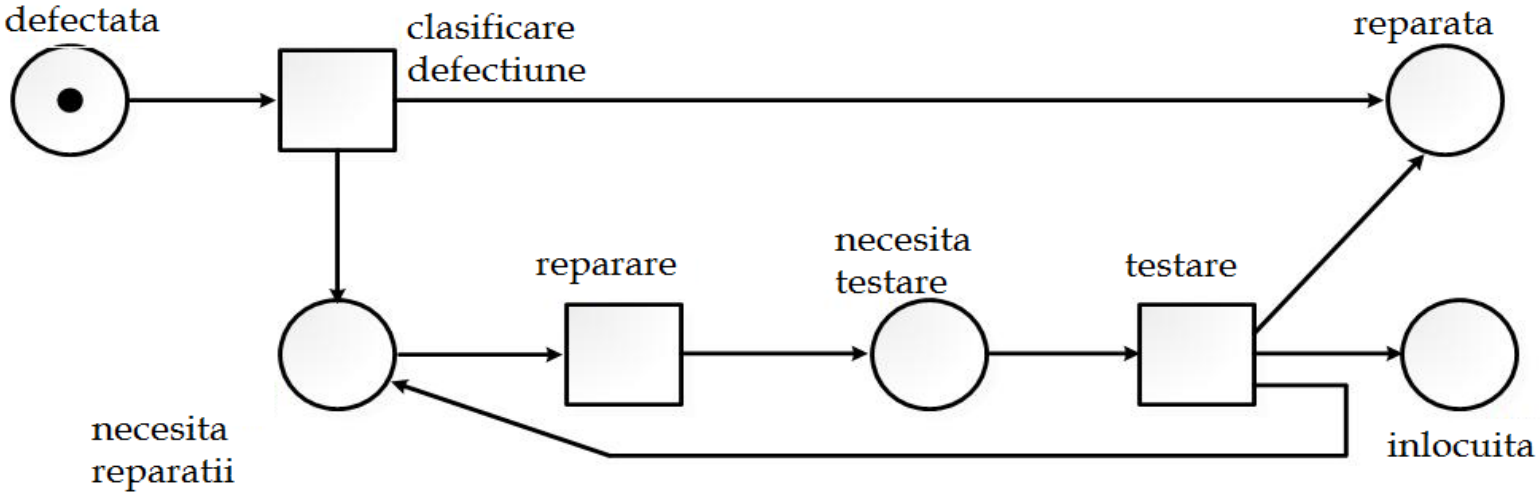
- Pentru a putea distinge intre ele jetoanele dintr-o pozitie, fiecarui jeton ii este asociat “o culoare” iar informația va fi reprezentată de ansamblul poziție-culoare.

- Într-o (RPC), fiecare tranziție poate fi executată în diferite maniere, reprezentate de diferitele culori de execuție ce sunt asociate tranziției.

- Executia unei tranzitii presupune ca:
  - numarul jetoanelor depuse (produse) depinde de valorile (culorile) jetoanelor consumate
  - valorile jetoanelor produse depind de valorile jetoanelor retrase

RPC – Exemplu

- Modelarea procesului de rezolvare a unei defectiuni tehnice intr-un departament de productie



Jetonul din starea *defectata* poate avea proprietatile:

- Natura defectiunii
- Identitatea componentei defectate
- Codul de localizare a componentei defectate
- Istoricul piesei (istoric al defectiunilor)

Retele Petri colorate (3)

- In RPC se pot asocia *preconditii* tranzitiilor, referitare la jetoanele care urmeaza a fi retrase (consumate). In acest caz, o tranzitie este executata doar daca exista cate un jeton in toate starile de intrare iar preconditiile sunt indeplinite.

- **Exemplu** de preconditie pentru tranzitia *clasificare defectiune:*

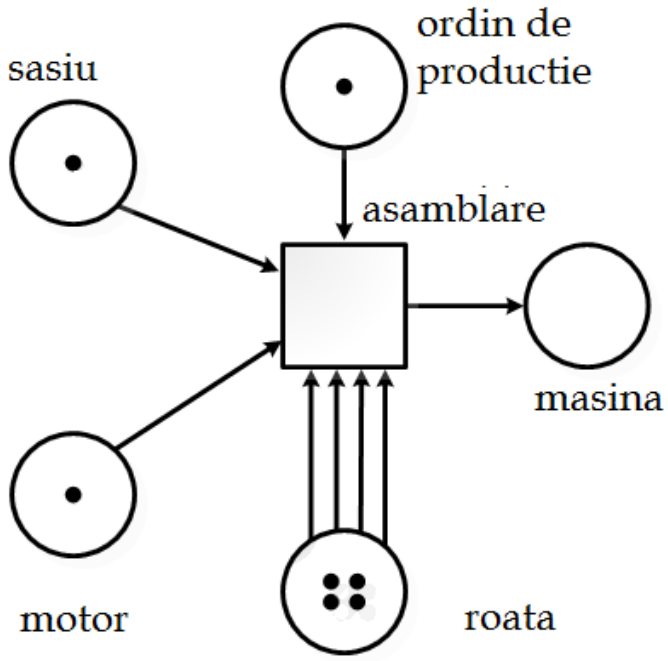
“ Valoarea jetonului care urmeaza a fi retras din starea *defectare* trebuie sa contina un cod de localizare valid*”.*

- **Consecinta** preconditiei:
  - Jetoanele care nu au aceasta proprietate valida nu sunt clasificate, ele ramanand in starea *defectata,* nefiind niciodata consumate de tranzitia *clasificare defectiune*

Retele Petri colorate (4)

- Alte utilizari ale preconditiilor:
  - ***Sincronizarea jetoanelor*** : o tranzitie este executata daca este disponibila o anumita combinatie de jetoane;

Exemplu: procesul de asamblare a unei masini



Retele Petri colorate (5)

Rezultatul utilizarii retelelor Petri colorate:

- reprezentare grafica mai simpla
- pentru fiecare tranzitie trebuie specificate elementele:
  - preconditiile – daca acestea exista, ele trebuiesc definite precis
  - valorile jetoanelor produse
  - numarul de jetoane depuse/iesire la fiecare executie a tranzitiei.

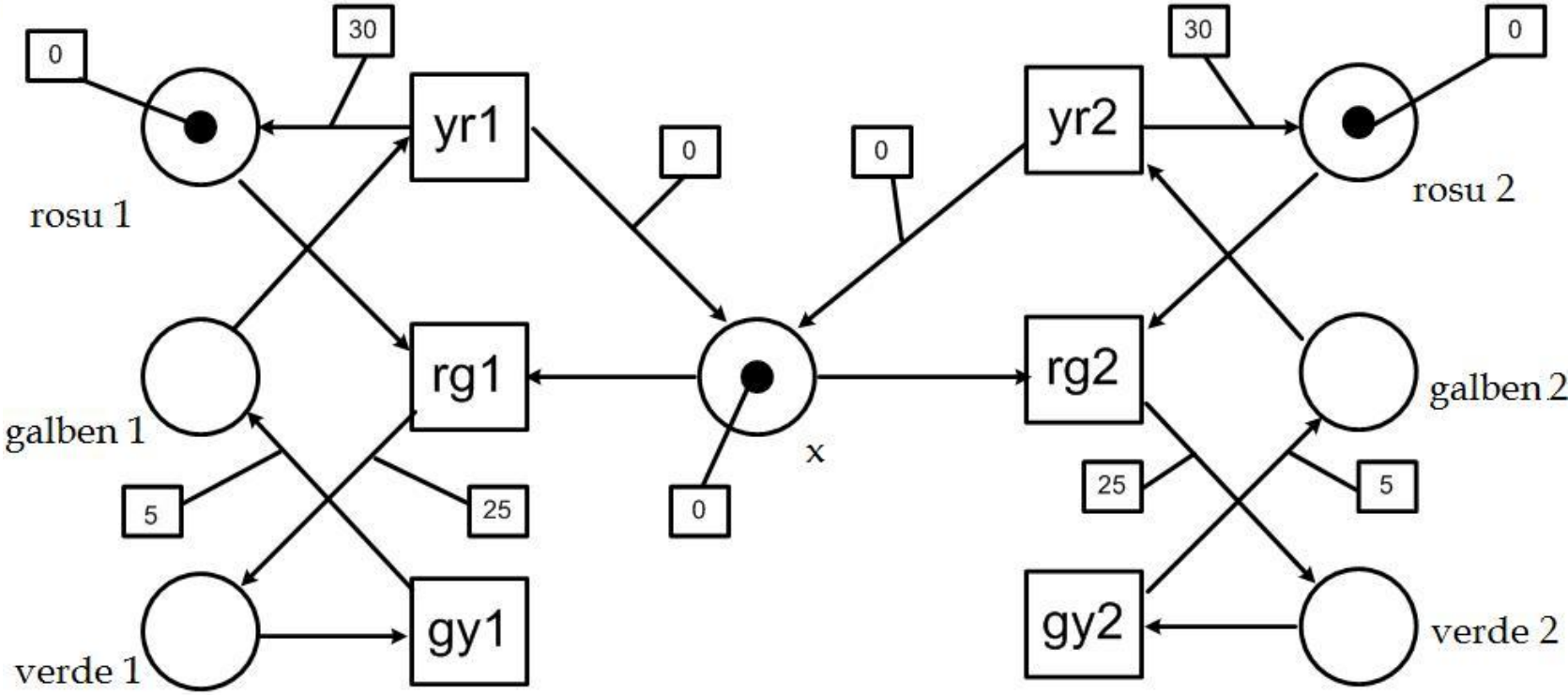
Retele Petri temporizate

- Sunt utilizate pentru a introduce informatii legate de performatele sistemului. Exemple:
  - numarul de masini dintr-o intersectie in unitatea de timp
  - timpul de executie a unei masini intr-o fabrica
  - capacitatea unei fabrici in unitatea de timp

- Cum se implementeaza timpul in retelele Petri?
  - se introduce notiunea de *timestamp* pe langa valoarea jetonului.
  - timestamp-ul indica din ce moment jetonul este disponibil

Retele Petri temporizate (2)

- Exemplu: sincronizarea a doua semafoare



- Obs. Prin adaugarea elementelor de *timp* modelului, nu doar am specificat durata diverselor tranzitii dar am *fortat* semafoarele sa afiseze verde alternativ

Retele Petri temporizate (3)

Executia unei tranzitii:

- are loc numai atunci cand toate jetoanele care vor fi retrase au un timestamp anterior sau egal cu momentul executiei tranzitiei
- pentru mai multe tranzitii cu acelasi timp de activare, se face o alegere nedeterminista
- executia unei tranzitii poate afecta activarea/dezactivarea altor tranzitii

Retele Petri ierarhizate

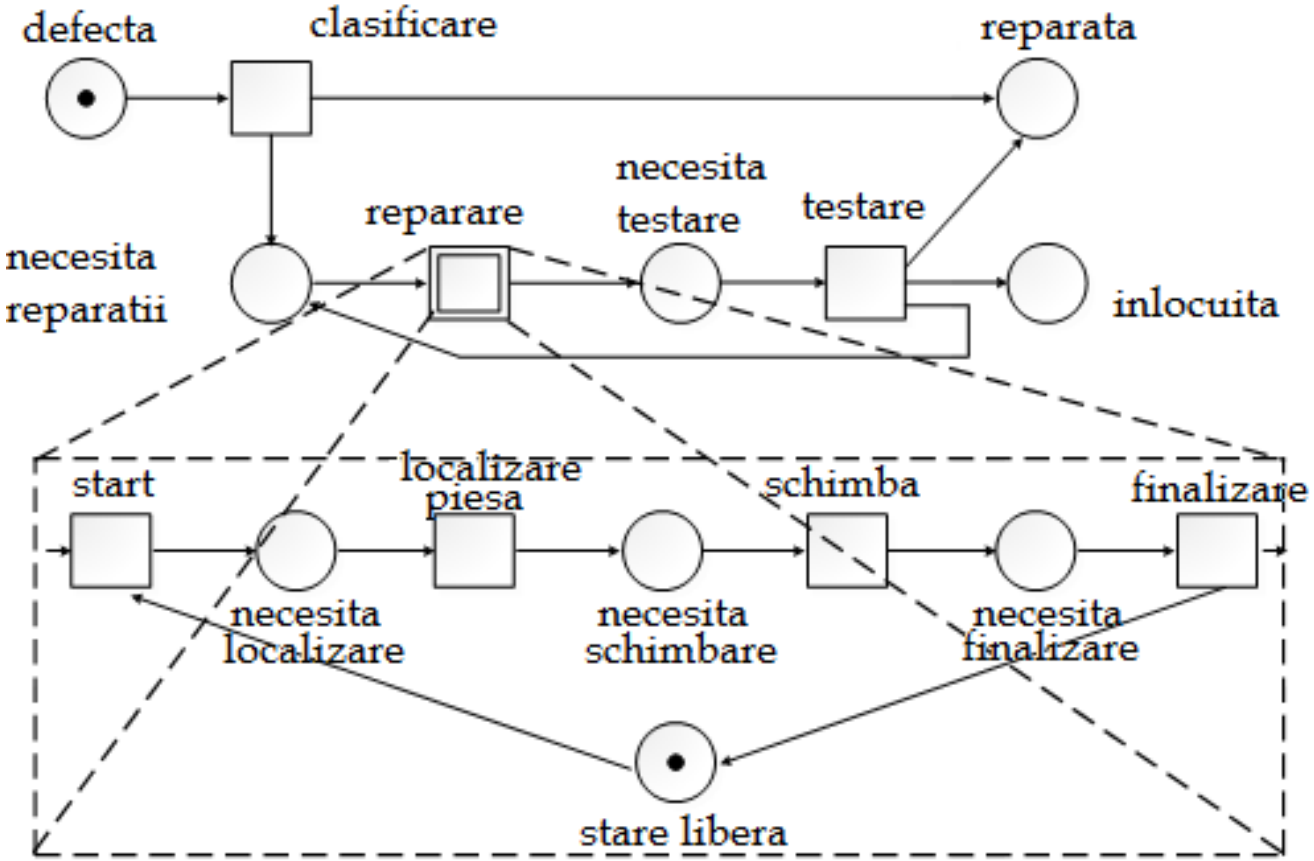
De ce am utiliza aceasta extensie a RP?

- deoarece RP de nivel inalt prezentate anterior tot nu reusesc sa ilustreze intr-un mod adecvat procesul care este modelat, rezultand in pierderea structurii procesului.

- Sunt utilizate structurile de tip proces (building block) –reprezentand o subretea ce cuprinde stari, tranzitii, arce si subprocese.

Retele Petri ierarhizate (2)

- Remodelarea procesului de rezolvare a unei defectiuni tehnice intr-un departament de productie.



Retele Petri ierarhizate (3)

- Actiunea de *reparare* nu mai este vazuta ca un bloc indivizibil ci ca un subproces care are urmatorii pasi:
  - start proces reparare
  - localizare piesa (dupa codul de localizare)
  - modificare piesa
  - finalizare proces

- Un proces poate lua doua forme:
  - subproces in cadrul unei structuri ierarhizate
  - proces alcatuit din mai multe subprocese

Retele Petri ierarhizate (4)

- Structurile ierahizate pot fi organizate:
  - top-down : prin descompunerea repetata se obtine o descriere ierarhizata
  - bottom-up : prin aceasta abordare se obtine descrierea intregului proces

- Strategii de obtinere a structurilor ierarhizate:
  - *divide-and-conquer :* presupune descompunerea procesului in subprocese mai putin complexe

Avantaj: posibilitatea reutilizarii unor procese definite anterior, scurtand timpul necesar modelarii proceselor complexe