

1. Sistemul de control al unui reactor chimic

1.1 Caietul de sarcini

În figura 1.1 este prezentată schema reactorului chimic. Acesta este reprezentat de un rezervor în care are loc reacția chimică și de un motor (M), care prin intermediul unei elici agită conținutul rezervorului. Servovanele Y1 și Y2 sunt folosite pentru a alimenta, respectiv goli, rezervorul.

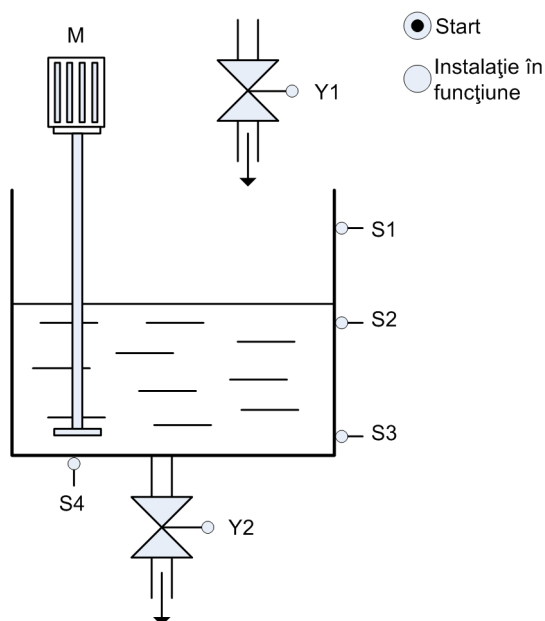


Fig. 1.1 Structura reactorului chimic

Pentru a realiza o doză de substanță chimică trebuie ca un muncitor să așeze manual

o cutie de reactiv în rezervor și să apese butonul de start. Mai departe automatizarea va alimenta rezervorul cu lichid până la nivelul senzorului S2. Alimentarea cu lichid se realizează prin deschiderea servovanei Y1. În continuare, motorul M agită conținutul rezervorului până când lichidul este omogen, fapt sesizat de senzorul S4. După omogenizare, automatizarea va alimenta rezervorul până la nivelul senzorului S1. Apoi aceasta va goli rezervorul prin deschiderea servovanei Y2. Ciclul de producție poate să reînceapă doar după ce senzorul S3 a sesizat că rezervorul este gol. Pe durata ciclului de producție trebuie să se aprindă indicatorul "Instalație în funcțiune".

1.2 Cerinte

1. Să se identifice stările și evenimentele definite în caietul de sarcini.
2. Pentru fiecare stare, să se stabilească modul de funcționare al actuatorilor (oprit/pornit).
3. Pentru fiecare eveniment să se stabilească senzorul care îl generează.
4. Folosind o structură de tip graf, să se proiecteze logica de control a automatizării.
5. Să se dezvolte o aplicație C care implementează logica de control.

2. Mediul software PNTOOL pentru simularea și analiza rețelelor Petri

PNTOOL (*Petri Net Toolbox*) [1] este un instrument software cu ajutorul căruia se poate realiza simularea și analiza rețelelor Petri. O descriere mult mai detaliată a modului de lucru cu PNTOOL se poate găsi în [2].

Mediul software PNTOOL utilizează o interfață grafică care devine activă după executarea comenzii din listingul 2.1 în linia de comandă a mediului MATLAB.

```
1 >>pntool
```

Listing 2.1 Pornirea interfeței grafice a mediului software PNTOOL

2.1 Construirea unui model

Pentru construirea unui model nou se selectează comanda *File->New Model* care va lansa o fereastră, ca în figura 2.1, unde se selectează tipul rețelei Petri. În cadrul laboratorului vor fi utilizate doar rețele Petri netemporizate.

După ce se selectează tipul rețelei Petri apare fereastra de desenare a rețelei (vezi fig. 2.2). În partea stângă a acestei ferestre se află butoanele de introducere a simbolurilor grafice. Butoanele au următoarea semnificație:

- 1) *Edit Objects*: permite modificarea obiectelor (poziții, arce, tranziții);
- 2) *Add Place*: realizează adăugarea unei poziții la model;
- 3) *Add Transition*: realizează adăugarea unei tranziții la model;
- 4) *Add Arc*: realizează adăugarea unui arc (de la poziție la tranziție sau de la tranziție la poziție) la model;
- 5) permite adăugarea de jetoane la pozițiile rețelei Petri.

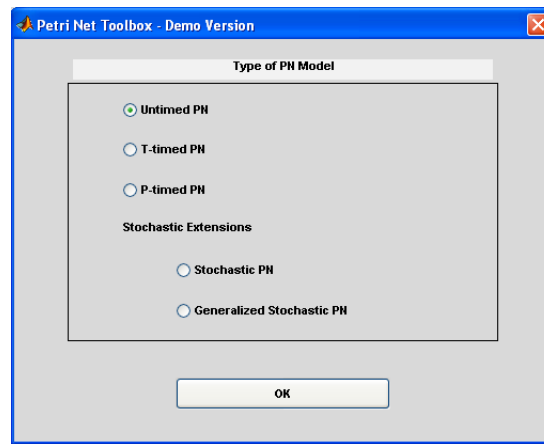


Fig. 2.1 Fereastră de selectare a modelului rețelei Petri.

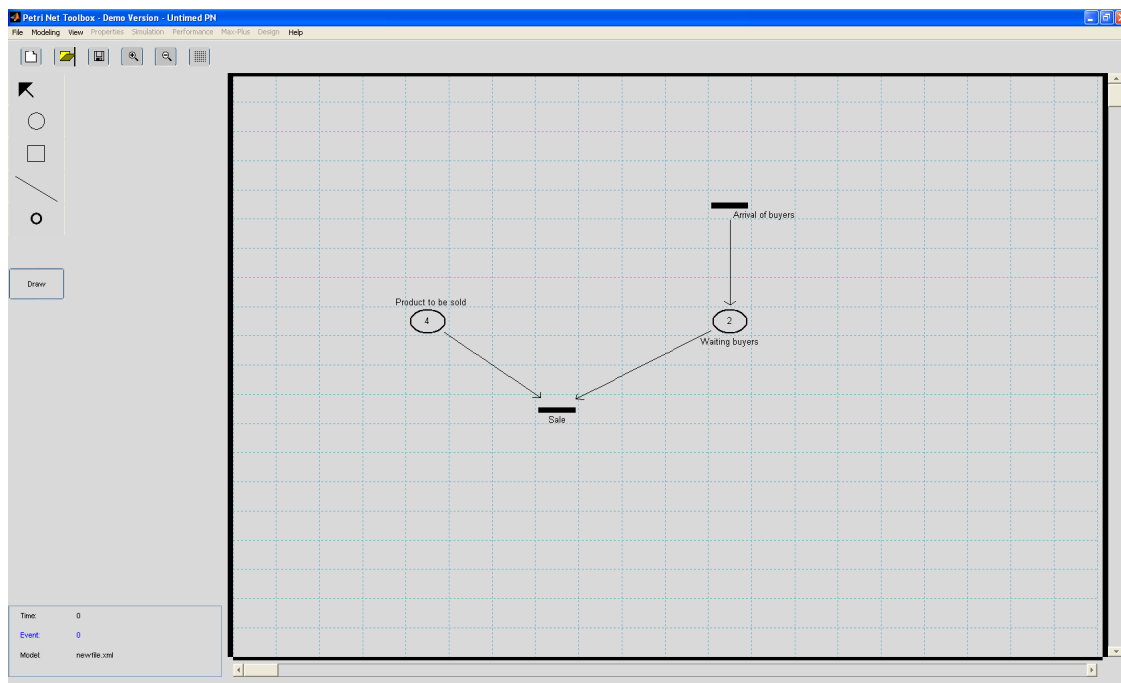


Fig. 2.2 Fereastră de desenare a rețelei Petri.

Pentru introducerea unei poziții sau a unei tranziții se apasă butonul *Add Place*, respectiv *Add Transition*, iar după aceea se face *click stânga* pe suprafața de lucru în locul unde se dorește plasarea simbolului grafic. În continuare, pentru mutarea sau modificarea proprietăților poziției, respectiv tranziției, se va utiliza meniul contextual care apare după ce se face *click dreapta* pe simbolul grafic respectiv. În fereastra de modificarea a proprietăților unei poziții se pot modifica, de fapt, numele poziției, culoarea, numărul de jetoane și capacitatea, iar în cazul unei tranziții se pot modifica: numele tranziției, mesajul care apare la executarea acesteia și culoarea.

Introducerea unui arc se realizează prin apăsarea butonului *Add Arc* și apoi făcând câte un *click stânga* pe nodul de plecare și pe cel de destinație. Nodurile fiind de fapt pozițiile sau tranzițiile rețelei Petri. De asemenea, pentru mutarea sau modificarea proprietăților arcului se utilizează meniul contextual care apare după efectuarea unui *click dreapta* pe arc. Proprietățile care pot fi modificate în cazul unui arc sunt: culoarea, tipul și ponderea. Pentru vizualizarea ponderilor arcelor trebuie să se dea comanda *View->Arc Weights*.

2.1.1 Setarea priorităților și/sau probabilităților pentru tranzițiile conflictuale

Două evenimente e_1 și e_2 sunt în conflict dacă pot apărea atât e_1 cât și e_2 dar nu pot apărea amândouă simultan. În figura 2.3 se prezintă o structură care modelează conflictul dintre evenimentele e_1 și e_2 materializate prin tranzițiile t_1 și respectiv t_2 . Ambele tranziții sunt validate dar numai una dintre ele se poate executa, execuția uneia conducând la invalidarea celei de-a doua [3].

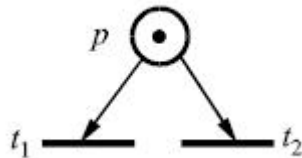


Fig. 2.3 Rețea Petri care modelează conflictul dintre evenimentele e_1 și e_2 materializate prin tranzițiile t_1 și respectiv t_2 .

În cazul tranzițiilor conflictuale, simulatorul PNT00L permite setarea unor priorități sau a unor probabilități de executare a tranzițiilor. Fereastra de rezolvare a conflictului dintre tranziții se accesează în felul următor: *Modeling->Resolution for Conflicting Transitions*. După ce se apasă butonul *Add*, pentru adăugarea unei noi reguli, sau butonul *Edit*, pentru modificarea unei reguli deja existente, simulatorul lansează o fereastră ca în figura 2.4. Câmpurile acestei ferestre se completează în felul următor:

- *Transitions*: reprezintă numele tranzițiilor pentru care se dorește rezolvarea conflictului (separate prin ",");
- *Values*: reprezintă probabilitatea sau prioritatea de producere a fiecărei tranziții din câmpul *Transitions*;
- *Type*: selectează modul de rezolvare a conflictului: prioritate sau probabilitate.

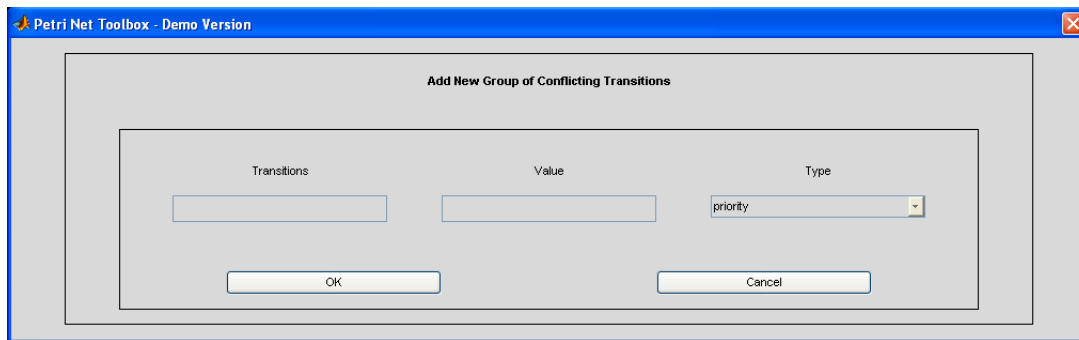


Fig. 2.4 Fereastră de stabilire a modului de rezolvare a conflictului dintre tranziții.

Dacă modul de rezolvare a conflictului utilizează probabilități atunci câmpul *Values* va conține k numere subunitare, separate prin ",", a căror sumă este 1. Fiecare număr subunitar reprezintă de fapt probabilitatea de producere a fiecărei tranziții din câmpul *Transitions* iar k reprezintă numărul de tranziții trecute în câmpul *Transitions*. Pentru situația în care se utilizează priorități pentru rezolvarea conflictului dintre tranziții, atunci câmpul *Values* va conține k numere întregi pozitive separate prin ",". Aceste numere reprezintă prioritatea tranzițiilor din câmpul *Transitions*. Se consideră că o tranziție este prioritară față de alta dacă valoarea din câmpul *Values* este mai mică.

2.1.2 Panoul de simulare

Panoul de simulare apare după ce se apasă butonul *Draw* (moment în care panoul de desenare dispare). Pentru a reveni în panoul de desenare se apasă butonul *Simulate*. După cum se poate observa și în figura 2.5, panoul de simulare conține următoarele comenzi:

- *Reset*: resetarea simulării (rețeaua Petri primește marcajul inițial);
- *Step*: realizează un pas de simulare (pe interfața grafică se poate observa tranziția care se execută și modificarea marcajelor);
- *Run Slow*: rulare lentă (permite vizualizarea tranzițiilor care se execută și modificarea marcajelor);
- *Run Fast*: rulare rapidă (nu permite vizualizarea tranzițiilor care se execută și modificarea marcajelor).

2.2 Analiza proprietăților comportamentale ale rețelilor Petri cu ajutorul mediului software PNTOL

Principalele proprietăți comportamentale ale unei rețele Petri sunt: *accesibilitatea*, *mărginirea*, *viabilitatea* și *reversibilitatea*. Acestea proprietăți comportamentale se pot determina dacă se analizează arborele de acoperire sau matricea de incidență și ecuația de stare [4].

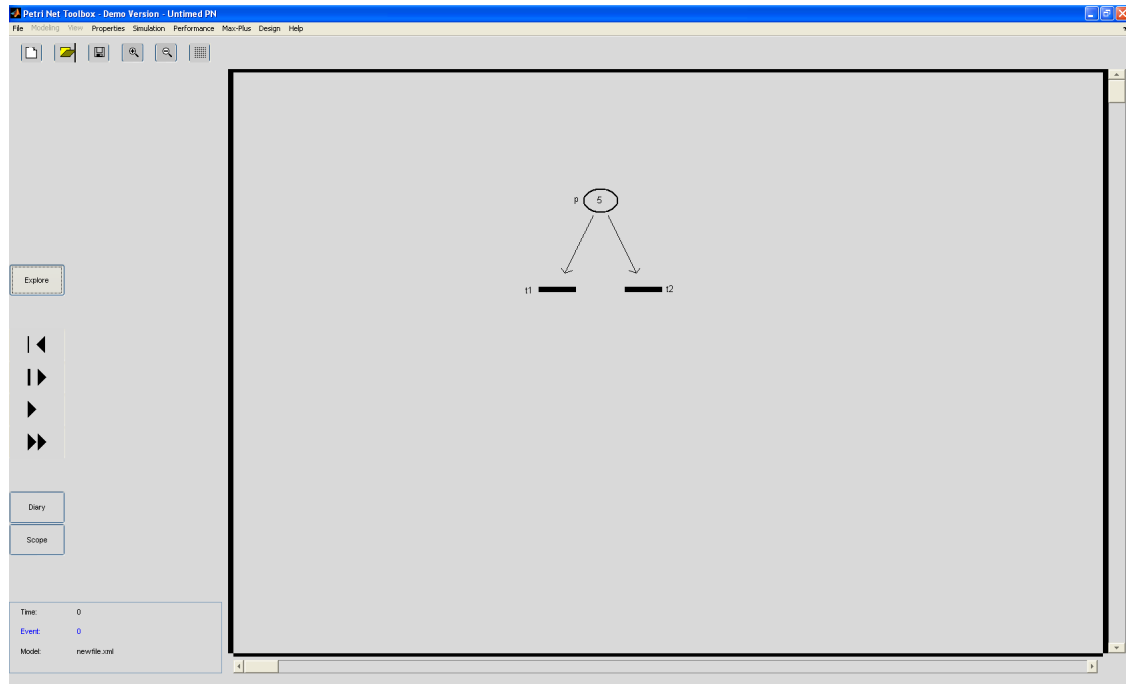


Fig. 2.5 Panoul de simulare.

2.2.1 Principalele proprietăți comportamentale ale unei rețele Petri

2.2.1.1 Accesibilitatea

Despre un marcaj M_n se poate spune că este accesibil din marcajul inițial M_0 dacă există o secvență de executări de tranziții care transformă M_0 în M_n [2]. Secvența de executări de tranziții se notează prin:

$$\sigma = M_0 t_1 M_1 t_2 M_2 \dots t_n M_n, \quad (2.1)$$

sau, dacă nu contează marcajele prin care trece rețeaua Petri pentru a se obține marcajul M_n :

$$\sigma = t_1 t_2 \dots t_n. \quad (2.2)$$

2.2.1.2 Mărginirea

O rețea Petri cu capacitate infinită este mărginită dacă numărul de jetoane din fiecare poziție nu depășește un număr finit k pentru orice marcaj accesibil din marcajul inițial M_0 . În limbaj matematic, o rețea Petri cu capacitate infinită este mărginită dacă:

$$M(p) \leq k \text{ pentru } \forall p \in P \text{ și } \forall M \in R(M_0), \quad (2.3)$$

unde, $R(M_0)$ reprezintă mulțimea de accesibilitate ce este corespunzătoare unui marcaj inițial dat.

2.2.1.3 Viabilitatea

O rețea Petri este viabilă dacă, indiferent de marcajul care a fost atins pornind din M_0 , este posibil ca, în continuare să fie executată orice tranziție t a rețelei. Până la executarea tranziției t se poate executa un număr finit de alte tranziții [2].

Un marcaj pentru care nicio tranziție a rețelei nu mai poate fi executată se numește *marcaj de deadlock*.

2.2.2 Arborele de acoperire

Pentru o rețea Petri (N, M_0) modificarea marcajelor (ca urmare a executării tranzițiilor) poate fi reprezentată sub forma unui arbore, denumit *arbore de acoperire* [3]. Construcția arborelui de acoperire se realizează folosind următorul algoritm [5]:

- 1) se stabilește M_0 ca rădăcină și se etichetează ca *marcaj nou*;
- 2) pentru fiecare marcaj M etichetat drept *marcaj nou*, se efectuează unul dintre următorii subpași:
 - 2.1) dacă M este identic cu un marcaj de pe drumul de la rădăcină la M , atunci M se etichetează ca *marcaj vechi*;
 - 2.2) dacă pentru marcajul M nu este nicio tranziție validată, atunci M se etichetează ca *marcaj de deadlock*;
 - 2.3) dacă pentru M există tranziții validate, atunci pentru fiecare tranziție validată t se efectuează următoarele etape:
 - se obține marcajul M' care rezultă din executarea tranziției t , pornind de la marcajul M ;
 - dacă pe drumul de la rădăcină la M exista un marcaj M'' astfel încât $M'(p) \geq M''(p)$ pentru orice poziție p și $M' \neq M''$ atunci $M'(p)$ se înlocuiește cu ω pentru fiecare poziție p în care avem inegalitatea $M'(p) \geq M''(p)$;
 - se introduce M' ca nod al arborelui de acoperire, se trasează un arc de la M la M' corespunzător tranziției t și se etichetează M' drept *marcaj nou*.

Arborele de acoperire al unei rețele Petri se obține, cu ajutorul mediului software PNTOL, în urma transmiterii comenzii *Properties -> Coverability Tree -> Graphic Mode*, iar marcajele M se obțin dacă se transmite comanda *Properties -> Coverability Tree -> Text Mode*.

2.2.3 Matricea de incidență și ecuația de stare

Se numește *matrice de incidență* a unei rețele Petri o matrice de dimensiune $n \times m$, unde n reprezintă numărul tranzițiilor, iar m numărul pozițiilor. Matricea de incidență este de forma $\mathbf{A} = [a_{ij}]$ ale cărei elemente sunt numere întregi:

$$a_{ij} = a_{ij}^+ - a_{ij}^-, \quad i = 1, \dots, n, \quad j = 1, \dots, m, \quad (2.4)$$

unde:

- $a_{ij}^+ = W(t_i, p_j)$ reprezintă ponderea arcului de la tranziția t_i , către poziția p_j ;

– $a_{ij}^- = W(p_j, t_i)$ reprezintă ponderea arcului de la poziția p_j , către tranziția t_i .

Matricea de incidență a unei rețele Petri se obține cu ajutorul mediului software PNT00L, în urma transmiterii comenzii: *Properties -> Incidence Matrix*.

Cunoscând matricea de incidență, se poate obține matricea de stare a rețelei Petri:

$$M_k = M_{k-1} + \mathbf{A}^T \mathbf{u}_k, \quad (2.5)$$

unde, M_k este marcajul care se obține din marcajul M_{k-1} dacă se execută tranziția t_i . Vectorul coloană \mathbf{u}_k , de dimensiune $n \times 1$, are toate elementele 0 cu excepția celui de al i -lea element care este 1 (corespunzător tranziției t_i).

Marcajul în care se ajunge după executarea unei secvențe de q tranziții (pornind din marcajul inițial) se poate obține cu ajutorul relației:

$$M_q = M_0 + \mathbf{A}^T \sum_{k=1}^q \mathbf{u}_k, \quad (2.6)$$

2.2.4 Teme aplicative

2.2.4.1 Arbore de accesibilitate și matrice de incidență

Să se construiască arborele de accesibilitate și matricea de incidență pentru rețelele Petri din figura 2.6.

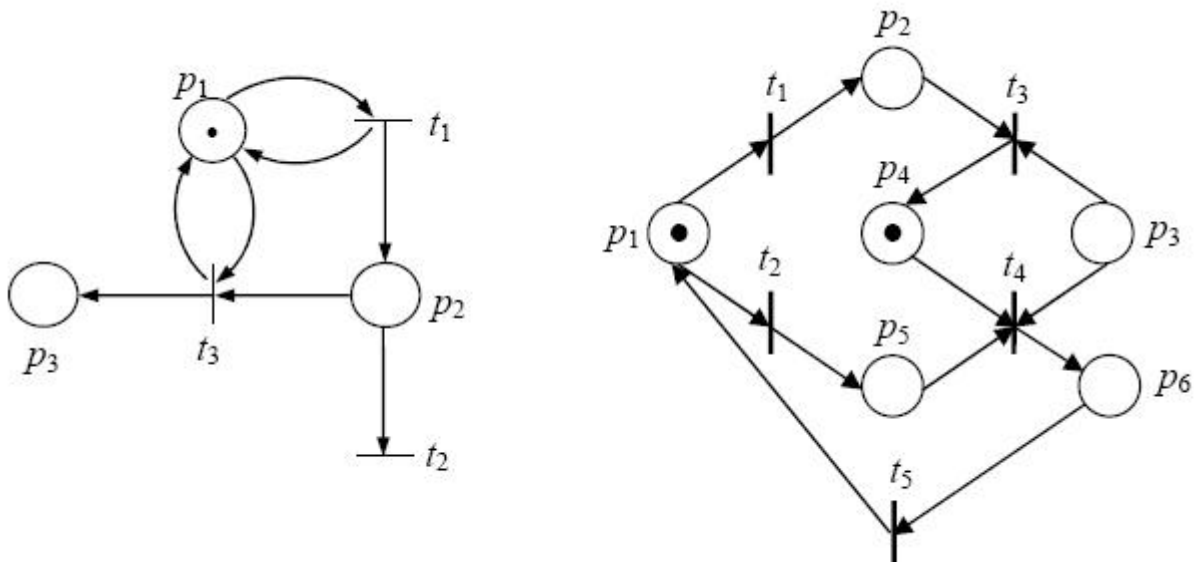


Fig. 2.6 Rețele Petri.

2.2.4.2 Sistem de calcul

Se consideră un sistem de calcul biprocesor, echipat cu două unități de disc D_1 , D_2 . Fiecare dintre cele două procesoare P_1 și P_2 execută câte o succesiune de sarcini, fiecare sarcină necesitând ambele discuri. Succesiunea de sarcini pentru procesorul P_1 necesită mai

întâi D_1 ; apoi, păstrând alocarea lui D_1 , se solicită alocarea lui D_2 ; în final D_1 și D_2 sunt eliberate simultan. Succesiune de sarcini pentru procesorul P_2 necesită mai întâi D_2 ; apoi, păstrând alocarea lui D_2 , se solicită alocarea lui D_1 ; în final D_1 și D_2 sunt eliberate simultan. O reprezentare schematizată a funcționării sistemului de calcul este dată în figura 2.7. La momentul inițial atât procesoarele cât și discurile sunt libere.

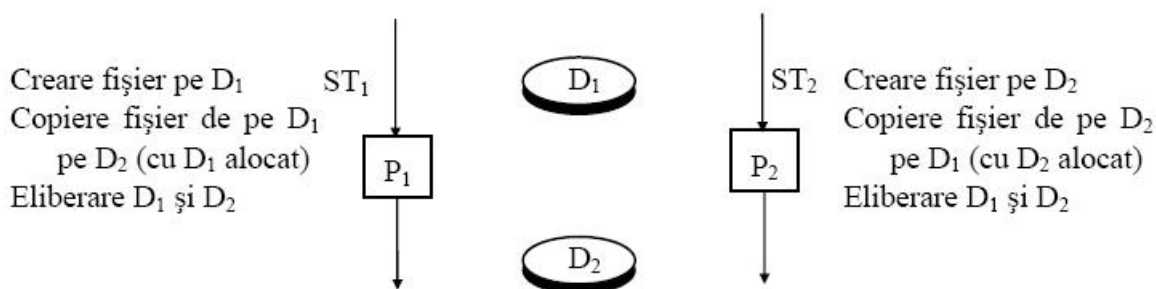


Fig. 2.7 Structură sistem calcul.

Cerințe:

- 1) să se construiască un model de tip rețea Petri pentru funcționarea sistemului de calcul, precizându-se semnificația fizică a pozițiilor și tranzițiilor;
- 2) să se construiască arborele de accesibilitate și matricea de incidență pentru modelul obținut la punctul anterior;
- 3) să se investigheze posibilitatea producerii fenomenului de deadlock în sistem; se va preciza marcajul rețelei Petri pentru care se ajunge în deadlock precum și secvențele de executări de tranziții care conduc la aceasta;
- 4) să se modifice modelul construit la punctul 1 astfel încât să se poată evita producerea fenomenului de deadlock.

3. Mediul software PetriNetExec pentru implementarea rețelelor Petri

PetriNetExec este o bibliotecă Java care permite integrarea rețelelor Petri în aplicațiile Java. Cu ajutorul bibliotecii PetriNetExec în aplicația Java se pot introduce poziții și tranziții; se pot lega acestea cu arce; se poate defini marcajul inițial; se pot trimite evenimente și se poate vedea fluxul de jetoane din rețea.

Pentru o înțelegere mai ușoară se va prezenta un exemplu de implementare a unei rețele Petri în PetriNetExec.

În figura 3.1 este prezentat modelul cu rețea Petri a comportamentului unui vehicul. Inițial, agentul este oprit (jetonul se afla în p_1) și trece în starea de accelerare (jetonul se află în p_2) dacă se produce evenimentul asociat tranziției t_1 (s-a apăsător pedala de accelerație). În continuare, vehiculul poate să încetinească (jetonul se află în p_3), dacă se produce evenimentul asociat tranziției t_3 (s-a apăsător pedala de frână), sau poate să mențină viteza constantă (jetonul se află în p_4) dacă se produce evenimentul asociat tranziției t_7 (vehiculul a atins viteza dorită).

Pentru început se crează o rețea Petri cu capacitate finită, care va fi denumită "Comportament_Vehicul" (vezi listing 3.1).

```
1 PetriNet net = new PetriNet("Comportament_Vehicul", PetriNet.Capacity.FINITE);
```

Listing 3.1 Crearea rețelei Petri

În continuare, se adaugă pozițiile rețelei Petri, conform listing-ului 3.2. Primul parametru reprezintă identificatorul poziției, cel de-al doilea parametru este un șir de caractere care poate fi utilizat la depanare, iar cel de-al treilea parametru reprezintă numărul de jetoane care se află în poziție la momentul inițial.

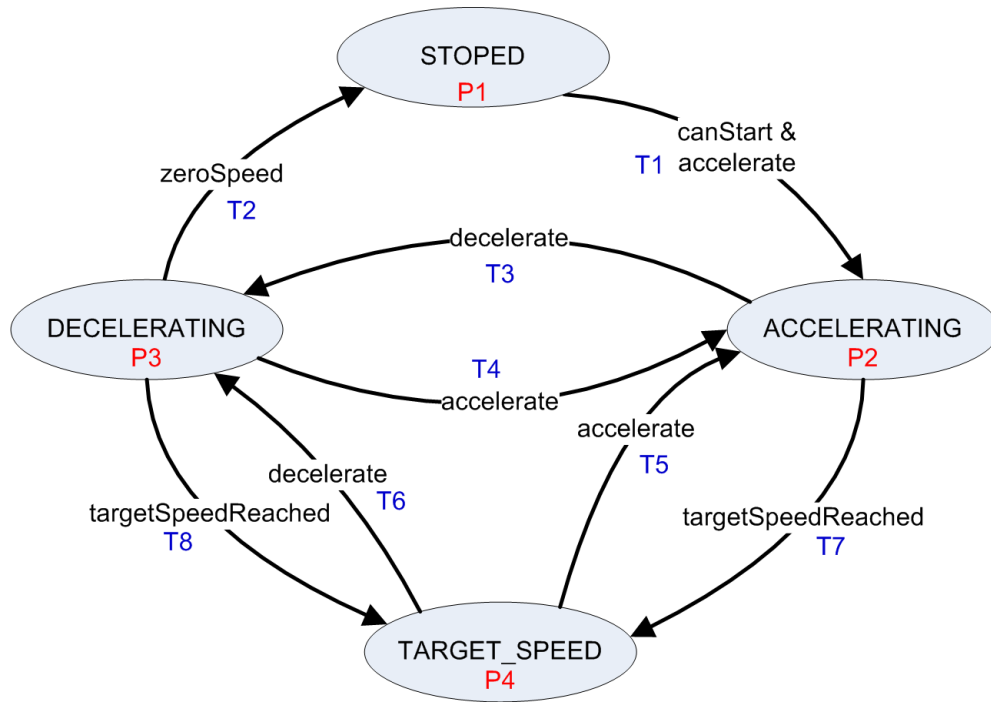


Fig. 3.1 Modelul cu rețea Petri al comportamentului unui vehicul.

```

1 String p1_ID = "P1";
  String p2_ID = "P2";
3 String p3_ID = "P3";
  String p4_ID = "P4";
5
  net.addPlace(p1_ID, "STOPED", 1);
7 net.addPlace(p2_ID, "ACCELERATING", 0);
  net.addPlace(p3_ID, "DECELERATING", 0);
9 net.addPlace(p4_ID, "TARGET_SPEED", 0);

```

Listing 3.2 Adăugarea pozițiilor la rețeaua Petri.

Tranzițiile sunt adăugate conform listing-ului 3.3. Primul parametru reprezintă identificatorul tranziției, cel de-al doilea parametru este un șir de caractere care poate fi utilizat la depanare, iar cel de-al treilea parametru reprezintă tipul tranziției (NORMAL - normală, SOURCE - sursă, SINK - destinație).

```

1 String t1_ID = "T1";
  String t2_ID = "T2";
3 String t3_ID = "T3";
  String t4_ID = "T4";
5 String t5_ID = "T5";
  String t6_ID = "T6";
7 String t7_ID = "T7";
  String t8_ID = "T8";
9
  net.addTransition(t1_ID, "canStart_and_accelarate", Transition.TransitionType.
    NORMAL);

```

```

11 net.addTransition(t2_ID, "zeroSpeed", Transition.TransitionType.NORMAL);
    net.addTransition(t3_ID, "decelerate", Transition.TransitionType.NORMAL);
13 net.addTransition(t4_ID, "accelerate", Transition.TransitionType.NORMAL);
    net.addTransition(t5_ID, "accelerate", Transition.TransitionType.NORMAL);
15 net.addTransition(t6_ID, "decelerate", Transition.TransitionType.NORMAL);
    net.addTransition(t7_ID, "targetSpeedReached", Transition.TransitionType.
        NORMAL);
17 net.addTransition(t8_ID, "targetSpeedReached", Transition.TransitionType.
    NORMAL);

```

Listing 3.3 Adăugarea tranzițiilor la rețeaua Petri.

Conectarea unui arc de la o poziție la o tranziție se face cu ajutorul metodei *connectToOutputTransition* (vezi listing 3.4) care conține următorii parametrii:

- tranziția la care se conectează arcul;
- identificatorul arcului;
- numele arcului;
- ponderea.

```

1 net.retrievePlace(p1_ID).connectToOutputTransition(net.retrieveTransition(
    t1_ID), null, "p1 to t1", 1);
    net.retrievePlace(p2_ID).connectToOutputTransition(net.retrieveTransition(
    t3_ID), null, "p2 to t3", 1);
3 net.retrievePlace(p2_ID).connectToOutputTransition(net.retrieveTransition(
    t7_ID), null, "p2 to t7", 1);
    net.retrievePlace(p3_ID).connectToOutputTransition(net.retrieveTransition(
    t2_ID), null, "p3 to t2", 1);
5 net.retrievePlace(p3_ID).connectToOutputTransition(net.retrieveTransition(
    t3_ID), null, "p3 to t3", 1);
    net.retrievePlace(p3_ID).connectToOutputTransition(net.retrieveTransition(
    t8_ID), null, "p3 to t8", 1);
7 net.retrievePlace(p4_ID).connectToOutputTransition(net.retrieveTransition(
    t5_ID), null, "p4 to t5", 1);
    net.retrievePlace(p4_ID).connectToOutputTransition(net.retrieveTransition(
    t6_ID), null, "p4 to t6", 1);

```

Listing 3.4 Conectarea arcelor de la poziții la tranziții.

Conectarea unui arc de la o tranziție la o poziție se face cu ajutorul metodei *connectToOutputPlace* (vezi listing 3.5) care conține următorii parametrii:

- poziția la care se conectează arcul;
- identificatorul arcului;
- numele arcului;
- ponderea.

```

    net.retrieveTransition(t1_ID).connectToOutputPlace(net.retrievePlace(p2_ID),
        null, "t1 to p2", 1);
2 net.retrieveTransition(t2_ID).connectToOutputPlace(net.retrievePlace(p1_ID),
    null, "t2 to p1", 1);

```

```

net.retrieveTransition(t3_ID).connectToOutputPlace(net.retrievePlace(p3_ID),
    null, "t3 to p3", 1);
4 net.retrieveTransition(t4_ID).connectToOutputPlace(net.retrievePlace(p2_ID),
    null, "t4 to p2", 1);
net.retrieveTransition(t5_ID).connectToOutputPlace(net.retrievePlace(p2_ID),
    null, "t5 to p2", 1);
6 net.retrieveTransition(t6_ID).connectToOutputPlace(net.retrievePlace(p3_ID),
    null, "t6 to p3", 1);
net.retrieveTransition(t7_ID).connectToOutputPlace(net.retrievePlace(p4_ID),
    null, "t7 to p4", 1);
8 net.retrieveTransition(t8_ID).connectToOutputPlace(net.retrievePlace(p4_ID),
    null, "t8 to p4", 1);

```

Listing 3.5 Conectarea arcelor de la tranziții la poziții.

În continuare, pentru verificare, se pot afișa pozițiile și tranzițiile utilizând metodele statice ale clasei *PetriNetUtils* (vezi listing 3.6).

```

PetriNetUtils.printPlaces(net, "Test print of the net places", true, true,
    PlacePrintingStyle.ALL, false);
2 PetriNetUtils.printTransitions(net, "Test print of the net transitions", true,
    true, TransitionPrintingStyle.ALL, false);

```

Listing 3.6 Afișarea pozițiilor și tranzițiilor din rețeaua Petri.

După construcția rețelei Petri trebuie realizat un generator de evenimente, cu ajutorul căruia se va efectua executarea tranzițiilor validate. De asemenea, generatorul trebuie anunțat ce rețea Petri îi ascultă evenimentele (vezi listing 3.7).

```

PetriNetEventGenerator eventGeneratorA = new PetriNetEventGenerator();
2 eventGeneratorA.addObserver(net.getEventObserver());

```

Listing 3.7 Adăugarea unui generator de evenimente la rețeaua Petri

Deoarece rețeaua Petri este folosită pentru modelarea comportamentului unei aplicații reale, trebuie implementat un mecanism de callback care să permită realizarea unor acțiuni, când se modifică marcajul rețelei Petri. În listing-ul 3.8 este prezentat callback-ul care se produce când numărul de jetoane din poziția p_1 scade sub 1.

```

net.retrievePlace(p1_ID).createExitCallback(1, new AgentPlaceAction(this.
    context){
2     @Override
    public void callback(PetriNet net, Place place)
4     {
        this.getAgentContext().agent.actionStopedExit();
6     }
    });
8
....
10
12 public void actionStopedExit(){
    System.out.println("STOPED exit");

```

```
}

```

Listing 3.8 Adăugarea unui mecanism de callback pentru poziția p_1 .

În mod similar, pentru poziții se pot defini următoarele callback-uri:

- *createEntryCallback*: când poziția primește un jeton;
- *createExitCallback*: când marcajul poziției scade sub o anumită valoare;
- *createEntryCallback*: când marcajul poziției crește peste o anumită valoare.

Pentru tranziții se pot defini următoarele callback-uri:

- *createEntryCallback*: când se execută tranziția.

La final, se poate utiliza metoda statică *checkNetwork* pentru verificarea rețelei Petri (vezi listing 3.9).

```
1 PetriNetUtils.checkNetwork(net, true);
```

Listing 3.9 Verificarea rețelei Petri.

În listing-ul 3.10 este prezentat modul de transmitere a evenimentelor către rețeaua Petri.

```
1 eventGeneratorA.pushEvent("T1", 100L);
```

Listing 3.10 Transmisie de evenimente către rețeaua Petri.

4. Sistem de fabricație cu trei puncte de lucru.

În figura 4.1 este prezentat un sistem de fabricație alcătuit din trei puncte de lucru, iar în tabelul 4.1 se află descrierea mărimilor de intrare și ieșire.

Tab. 4.1 Descrierea mărimilor de intrare și ieșire ale sistemului de fabricație cu trei puncte de lucru.

| Mărime | Descriere |
|---------------|---|
| Y_1, Y_2 | comanda revenire/avans piston P_1 |
| Y_3, Y_4 | comanda revenire/avans piston P_2 |
| Y_5, Y_6 | comanda revenire/avans piston P_3 |
| Y_7, Y_8 | comanda revenire/avans piston P_4 |
| Y_9, Y_{10} | comanda revenire/avans piston P_5 |
| Y_{11} | comanda evacuare piesă presată |
| Y_{12} | pornire pulverizator adeziv |
| Y_{13} | comanda evacuare piesă pulverizată cu adeziv |
| Y_{14} | încălzire piese pentru activare adeziv |
| BCI | buton confirmare încărcare magazie de piese M |
| S_1, S_2 | senzori extremitați piston P_1 |
| S_3, S_4 | senzori extremitați piston P_2 |
| S_5, S_6 | senzori extremitați piston P_3 |
| S_7, S_8 | senzori extremitați piston P_4 |
| S_9, S_{10} | senzori extremitați piston P_5 |
| S_{11} | senzor prezență piesă la punctul de presare |
| S_{12} | senzor de temperatură |
| S_{13} | senzor prezență piesă la punctul de pulverizare |
| S_{14} | senzor trecere piese către punctul de activare adeziv |
| AL | alarmă magazie de piese goală |

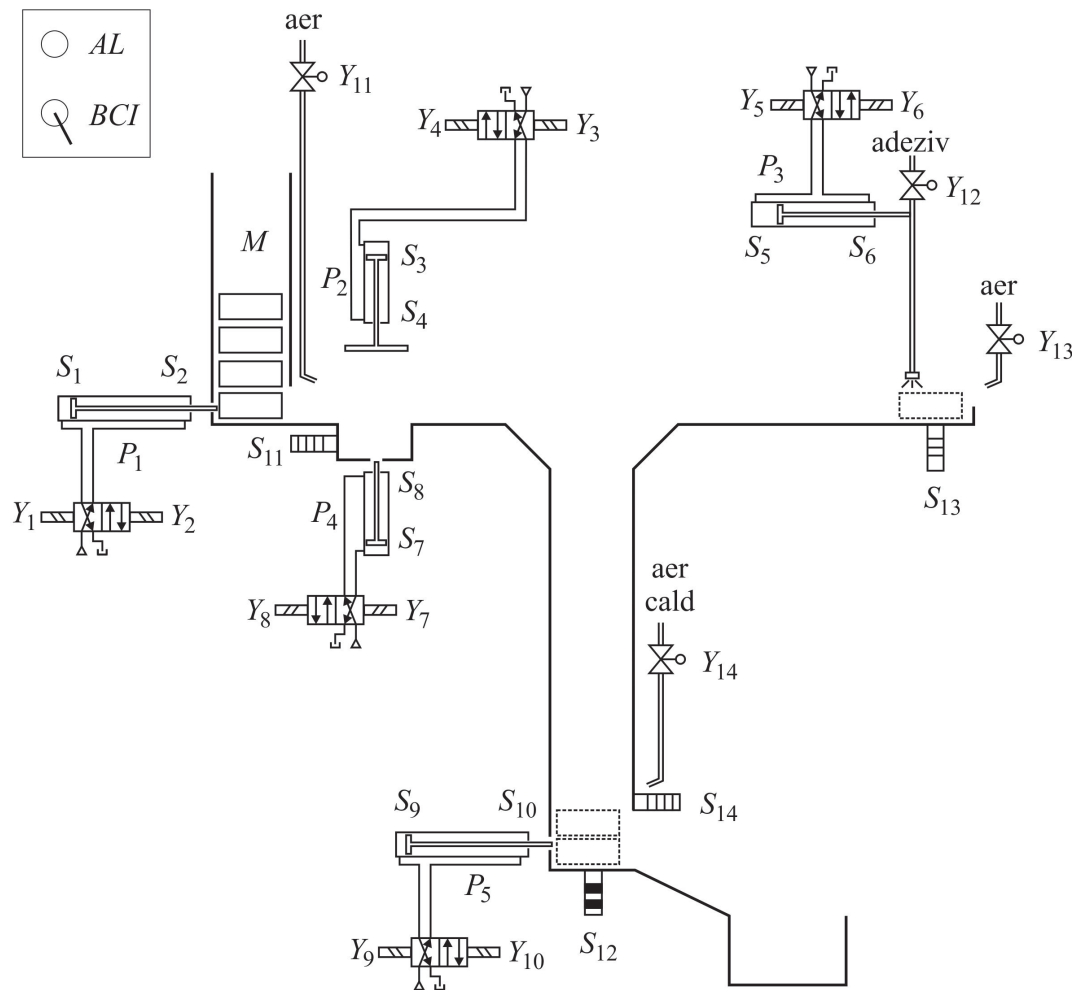


Fig. 4.1 Schema generală a sistemului de fabricație cu trei puncte de lucru.

Primul punct de lucru realizează pulverizarea cu adeziv a unei piese. Această operație se realizează prin transmiterea comenzii Y_6 care conduce la avansul pistonului P_3 . În același timp trebuie pornit și pulverizatorul de adeziv cu ajutorul comenzii Y_{12} . După finalizarea procesului de pulverizare adeziv (a fost activat senzorul S_6) se oprește pulverizatorul prin dezactivarea comenzii Y_{12} și se retrage pistonul P_3 prin transmiterea comenzii Y_5 până când se activează senzorul S_5 . La final piesa este evacuată cu ajutorul jetului de aer care se obține în urma activării comenzii Y_{13} .

Cel de-al doilea punct de lucru realizează ștanțarea unei piese cu ajutorul pistonului P_2 . Piesa este încărcată din magazia M în punctul de lucru cu ajutorul pistonului P_1 prin transmiterea comenzii Y_2 . Trebuie să se considere și situația în care magazia de piese M este goală. În momentul în care se activează comanda Y_2 , pistonul P_1 începe să împingă o piesă din magazia M către punctul de lucru. Dacă piesa ajunge la punctul de lucru (se generează evenimentul S_{11}) înseamnă că magazia conținea cel puțin o piesă. Dacă nu se generează acest eveniment și pistonul P_1 ajunge la capătul de cursă (s-a produs evenimentul S_2) înseamnă că magazia nu conținea piese. În această situație se aprinde indicatorul de alarmă AL și se așteaptă încărcarea magaziei cu piese. După ce utilizatorul a încărcat magazia M , acesta trebuie să apese butonul BCI , moment în care pistonul P_1 reîncepe procesul de împingere

al piesei. După ce piesa a fost adusă în locașul pentru ștanțare, se transmite comanda Y_4 pentru ca pistonul P_2 să înceapă coborârea. Se va considera că piesa este ștanțată în momentul în care s-a generat evenimentul asociat senzorului S_4 . În continuare se așteaptă retragerea pistonului P_2 (după ce s-a transmis comanda Y_4). Pentru a descărca piesa din punctul de ștanțare se acționează pistonul P_4 prin transmiterea comenzii Y_8 și în același timp se pornește jetul de aer cu ajutorul vanei comandate de mărimea de ieșire Y_{11} . Procedura de evacuare piesă se finalizează prin retragerea completă a pistonului P_4 .

Ultimul punct de lucru realizează lipirea celor două piese prelucrate anterior. Activarea adezivului se realizează prin încălzirea pieselor cu ajutorul unui jet de aer cald controlat de vana Y_{14} . Senzorul de temperatură S_{12} este cel care sesizează dacă ansamblul pieselor a ajuns la temperatura dorită. Pentru evacuarea piesei finale se utilizează pistonul P_5 . Acesta primește comanda Y_{10} până când piesa este evacuată (s-a generat evenimentul S_{10}), după care primește comanda Y_9 până când se retrage pistonul.

Modul de operare al instalației este următorul:

- utilizatorul uman așează o piesă la punctul de aplicare adeziv, aspect care este sesizat cu ajutorul senzorului S_{13} ;
- se pornește în paralel procesul de aplicare adeziv și cel de ștanțare (care presupune în prealabil încălzirea unei piese);
- când procesul de activare adeziv este finalizat piesa este descărcată (aspect sesizat de senzorul S_{14});
- procesul de ștanțare poate să realizeze descărcarea piesei numai după ce s-a finalizat pasul anterior (s-a descărcat piesa pe care s-a pulverizat adeziv) și s-a apăsă butonul BCI ;
- se pornește procesul de activare adeziv și se descarcă piesa finală.

4.0.1 Cerințe:

1. Să se proiecteze folosind o rețea Petri logica de control a automatizării.
2. Să se testeze rețeaua Petri folosind unealta PNTTOOL.

5. Sistem de fabricație cu resurse partajate.

În figura 5.1 este prezentat un sistem de fabricație cu resurse partajate, iar în tabelul 5.1 se află descrierea mărimilor de intrare și ieșire.

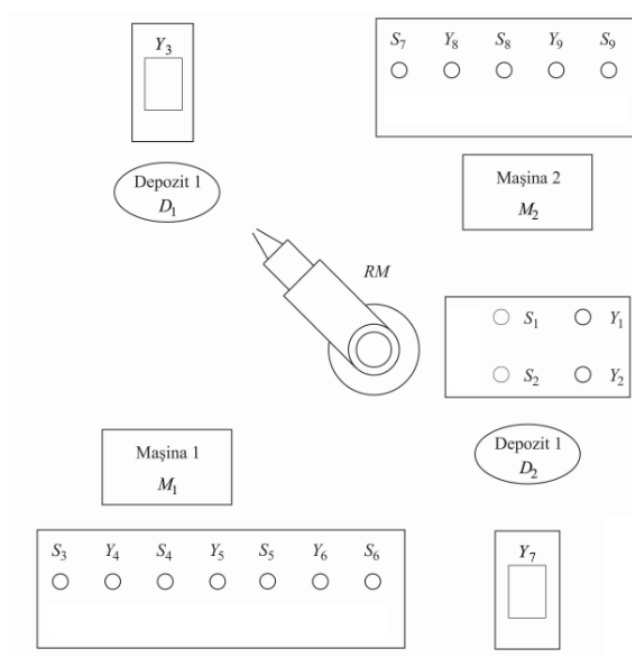


Fig. 5.1 Schema generală a sistemului de fabricație cu resurse partajate.

Robotul mobil RM este o resursă partajată deoarece este folosit pentru a încărca piese din depozitul D_1 pe mașina unealtă M_1 , respectiv pentru a încărca piese din depozitul D_2 pe mașina unealtă M_2 . La un moment dat de timp robotul mobil poate să transporte

Tab. 5.1 Descrierea mărimilor de intrare și ieșire ale sistemului de fabricație cu resurse partajate.

| Mărimie | Descriere |
|---------|---|
| Y_1 | RM încarcă o piesă pe mașina M_1 |
| Y_2 | RM încarcă o piesă pe mașina M_2 |
| Y_3 | numărul de paleți din depozitul D_1 |
| Y_4 | mașina M_1 prelucrează o piesă |
| Y_5 | mașina M_1 așteaptă descărcarea |
| Y_6 | mașina M_1 descarcă piesa prelucrată |
| Y_7 | numărul de paleți din depozitul D_2 |
| Y_8 | mașina M_2 prelucrează o piesă |
| Y_9 | mașina M_2 descarcă piesa prelucrată |
| S_1 | comanda RM pentru încărcarea unei piese pe mașina M_1 |
| S_2 | comanda RM pentru încărcarea unei piese pe mașina M_2 |
| S_3 | finalizarea încărcării unei piese pe mașina M_1 |
| S_4 | finalizarea prelucrării unei piese pe mașina M_1 |
| S_5 | comanda descărcării piesei prelucrate pe mașina M_1 |
| S_6 | finalizarea descărcării piesei prelucrate pe mașina M_1 |
| S_7 | finalizarea încărcării unei piese pe mașina M_2 |
| S_8 | finalizarea prelucrării unei piese pe mașina M_2 |
| S_6 | finalizarea descărcării piesei prelucrate pe mașina M_2 |

doar o singură piesă. Senzorul S_1 este folosit pentru a comanda robotul mobil să mute o piesă din depozitul D_1 pe mașina unealtă M_1 . Pe durata mutării trebuie ca indicatorul Y_1 să fie activat. În mod similar, senzorul S_2 și indicatorul Y_2 sunt folosiți pentru a realiza mutarea unei piese din depozitul D_2 pe mașina unealtă M_2 .

Pe perioada procesului de prelucrare piesele sunt transportate cu ajutorul unor paleți. Piesele brute sunt încărcate automat pe un palet și sunt stocate în depozitul D_1 . În total sistemul de fabricație conține 5 paleți. La momentul initial, fiecare palet conține câte o piesă și este stocat în depozitul D_1 .

Robotul mobil va începe procesul de mutare al unei piese din depozitul D_1 pe mașina unealtă M_1 doar dacă următoarele condiții sunt îndeplinite simultan:

- în depozitul D_1 se găsește cel puțin o piesă;
- mașina unealtă M_1 este disponibilă (a descărcat ultima piesă prelucrată);
- robotul mobil este disponibil;
- se apasă butonul S_1 .

Mai departe, senzorul S_3 sesizează momentul în care robotul mobil RM a terminat de încărcat o piesă pe mașina unealtă M_1 . Mașina unealtă începe să prelucreze automat piesa primită, perioadă în care indicatorul Y_4 este activat. Finalizarea acestui proces este sesizată de senzorul S_4 . În continuare, mașina unealtă așteaptă comanda de descărcare a piesei, perioadă în care se activează indicatorul Y_5 . Utilizatorul este cel care generează comanda de

descărcare prin apăsarea butonului S_5 . Mașina unealtă M_1 va realiza descărcarea doar dacă în depozitul D_2 sunt mai puțin de două piese. Pe perioada descărcării indicatorul Y_6 este activat iar finalizarea acesteia este sesizată de senzorul S_6 .

Robotul mobil va începe procesul de mutare al unei piese din depozitul D_2 pe mașina unealtă M_2 doar dacă următoarele condiții sunt îndeplinite simultan:

- în depozitul D_2 se găsește cel puțin o piesă;
- mașina unealtă M_2 este disponibilă (a descărcat ultima piesă prelucrată);
- robotul mobil este disponibil;
- se apasă butonul S_2 .

Senzorul S_7 sesizează momentul în care o piesă este încărcată pe mașina unealtă M_2 . Mai departe aceasta prelucrează piesa și activează indicatorul Y_8 . Finalizarea prelucrării este sesizată de senzorul S_8 . După finalizarea unei piese, mașina unealtă M_2 o va descărca direct, perioadă în care indicatorul Y_9 este activ. Senzorul S_9 sesizează momentul în care se termină descărcarea.

După ce mașina unealtă descarcă o piesă, paletul este automat reîncărcat cu o piesă brută și stocat în depozitul D_1 .

5.0.1 Cerințe:

1. Să se proiecteze folosind o rețea Petri logica de control a automatizării.
2. Să se testeze rețeaua Petri folosind unealta PNTOL.

6. Controlul supervizat al roboților autonomi dintr-o incintă.

În figura 6.1 este prezentată schema incintei unde operează roboții autonomi, iar în tabelul 6.1 se află descrierea mărimilor de intrare și ieșire.

Tab. 6.1 Descrierea mărimilor de intrare și ieșire ale sistemului de control supervizat al roboților.

| Mărime | Descriere |
|----------|---|
| $Gara_j$ | numărul de roboți din spațiul de garare |
| C_1 | numărul de roboți din camera 1 |
| C_2 | numărul de roboți din camera 2 |
| C_3 | numărul de roboți din camera 3 |
| Y_1 | semafor trecere din garaj în camera 1 |
| Y_2 | semafor trecere din camera 1 în camera 2 |
| Y_3 | semafor trecere din camera 3 în camera 2 |
| Y_4 | semafor trecere din camera 2 în camera 3 |
| Y_5 | semafor trecere din camera 3 în camera 1 |
| Y_6 | semafor trecere din camera 2 în garaj |
| B_1 | buton pentru comanda deplasării unui robot din garaj în camera 1 |
| B_2 | buton pentru comanda deplasării unui robot din camera 1 în camera 2 |
| B_3 | buton pentru comanda deplasării unui robot din camera 3 în camera 2 |
| B_4 | buton pentru comanda deplasării unui robot din camera 2 în camera 3 |
| B_5 | buton pentru comanda deplasării unui robot din camera 3 în camera 1 |
| B_6 | buton pentru comanda deplasării unui robot din camera 2 în garaj |

La momentul inițial în garaj se găsesc 6 roboți. Utilizatorul uman poate să comande roboții să se deplaseze dintr-o camera în alta. Comanda de mutare se realizează prin apăsarea unuia dintre butoanele $B_1 \dots B_6$ (vezi tabelul 6.1).

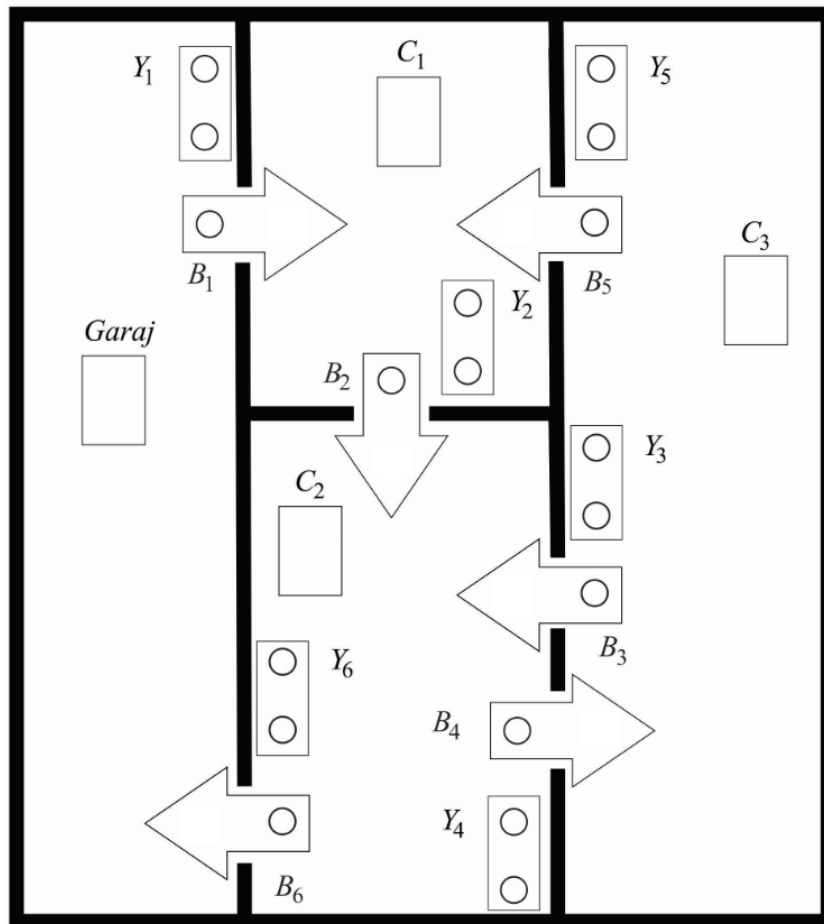


Fig. 6.1 Schema incintei unde operează roboții autonomi.

Incinta unde operează roboții are următoarele constrângeri:

- în camera 1 pot intra maxim 2 roboți;
- numărul total de roboți din camerele 1 și 2 nu poate să fie mai mare de 4;
- numărul total de roboți din camerele 2 și 3 nu poate să fie mai mare de 3.

Indicatoarele *Garaj*, C_1 , C_2 și C_3 sunt folosite pentru a afișa numărul de roboți din camera respectivă. Indicatoarele $Y_1 \dots Y_6$ sunt activate în momentul în care nu se poate intra în camera respectivă.

6.0.1 Cerințe:

1. Să se proiecteze folosind o rețea Petri logica de control a automatizării.
2. Să se testeze rețeaua Petri folosind unealta PNTOL.

7. Controlul unei centrale termice pe gaz

7.1 Caiet de sarcini

În figura 7.1 este prezentată structura unei centrale termice pe gaz. Tabelul 7.1 conține descrierea mărimilor de intrare și de ieșire ale sistemului.

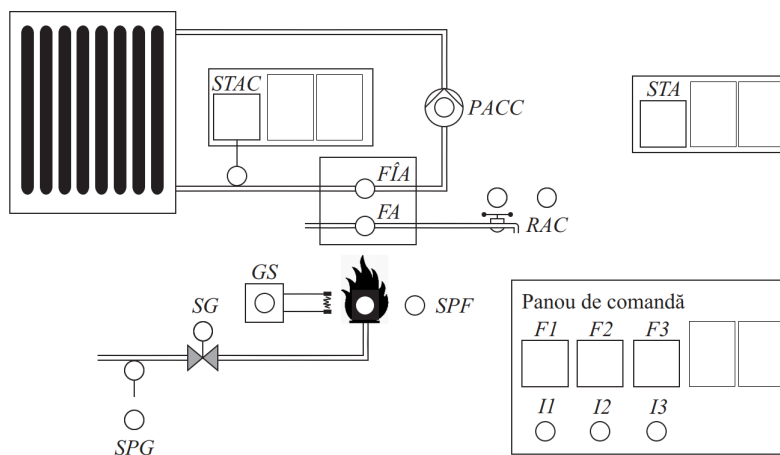


Fig. 7.1 Structura unei centrale termice pe gaz.

7.1.1 Funcționarea sistemului de încălzire al centralei termice

Sistemul de încălzire este folosit pentru a încălzi apa care circulă prin circuitul de căldură sau prin robinetul de apă caldă. Acesta are următoarele componente:

- arzătorul;

Tab. 7.1 Descrierea mărimilor de intrare și ieșire ale sistemului de control al centralei termice.

| Mărime | Descriere |
|--------|---|
| PACC | pompa de apă a circuitului de căldură |
| FÎA | funcția de încălzire ambientală |
| FA | funcția de încălzire apă |
| SG | servovană gaz |
| GS | generator scânteie |
| I1-3 | indicatoare ale panoului de comandă |
| RAC | stare robinet apă caldă |
| STAC | senzor temperatură apă din circuitul de căldură |
| SPG | senzor presiune gaz |
| SPF | senzor prezență flacără |
| STA | senzor temperatură ambientală |
| F1-3 | butoane ale panoului de comandă |

- generatorul de scânteie pentru a aprinde flacăra arzătorului;
- senzorul de prezență flacără;
- servovana de gaz;
- senzorul de presiune de pe conducta de gaz.

Se consideră că la momentul inițial sistemul de încălzire al centralei termice este oprit. Comanda START SISTEM ÎNCĂLZIRE este luată în considerare doar dacă senzorul SPG confirmă că pe conducta de alimentare cu gaz presiunea este la valoarea nominală. Mai departe sistemul de încălzire va deschide servovana de gaz (SG) și va porni generatorul de scânteie (GS). Dacă în 3 secunde nu apare flacăra, fapt sesizat de senzorul SPF, centrala termică intră în avarie și va afișa valoarea 99 pe panoul de comandă și se activează indicatorul I3. Apăsarea butonului F1 va readuce sistemul în starea de așteptare a comenzii de START SISTEM ÎNCĂLZIRE. Dacă senzorul SPF detectează flacăra în mai puțin de 3 secunde, atunci se oprește generatorul de scânteie și se menține deschisă servovana de gaz. Aceasta trebuie să se închidă dacă a apărut comanda STOP SISTEM ÎNCĂLZIRE, iar în această situație ciclul se poate relua. De asemenea, servovana SG trebuie să se închidă dacă pe parcursul funcționării senzorul SPG sesizează că presiunea de pe conducta de gaz a scăzut sau dacă flacăra s-a stins (SPF). În acest caz centrala termică intră din nou în starea de avarie și va afișa valoarea 98 pe panoul de comandă, respectiv va aprinde indicatorul I3. Din această stare de avarie se poate ieși doar dacă se apasă butonul F1.

7.1.2 Funcția de încălzire apă

În această situație se consideră că centrala termică este folosită doar pentru a încălzi apa pentru robinet. Deschiderea robinetului de apă (RAC) va genera comanda START SISTEM ÎNCĂLZIRE. Închiderea robinetului de apă va genera comanda STOP SISTEM

ÎNCĂLZIRE. Indicatorul FA trebuie să fie activ atâta timp cât funcția de încălzire este pornită.

7.1.3 Funcția de încălzire ambientală

În această situație se consideră că centrala termică are doar funcția de încălzire ambientală. Temperatura de referință a apei din circuitul de căldură este setată implicit la valoare de TEMPERATURĂ REFERINȚĂ APĂ DIN CIRCUIT = 60°C. Centrala termică are două etape distincte după cum urmează: verificarea temperaturii din circuitul de căldură și încălzirea apei din circuitul de căldură.

În prima etapă, trebuie ca centrala la fiecare 30 de secunde să pornească pompa de apă (PACC). Dacă timp de 10 secunde temperatura apei din circuitul de căldură nu scade sub valoarea $0.9 * \text{TEMPERATURĂ REFERINȚĂ}$, atunci pompa se oprește. În această situație pompa va reporni la următorul ciclu de 30 de secunde. Temperatura apei din circuitul de căldură este măsurată cu ajutorul senzorului STAC. Dacă temperatura apei scade sub valoarea $0.9 * \text{TEMPERATURĂ REFERINȚĂ}$ atunci se va activa etapa a doua.

În cea de-a doua etapă trebuie să se realizeze încălzirea apei din circuitul de căldură. În această etapă trebuie ca pompa de apă PACC, respectiv indicatorul FÎA, să fie tot timpul activi. De asemenea la început se transmite comanda START SISTEM ÎNCĂLZIRE. În prima etapă se revine doar în momentul în care temperatura apei din circuitul de căldură depășește valoare $1.1 * \text{TEMPERATURĂ REFERINȚĂ}$.

Dacă la un moment dat de timp se detectează o stare de avarie precum cele prezentate în secțiunea 7.1.1 ambele etape sunt dezactivate. În momentul în care se resetează sistemul prin apăsarea butonului F1, trebuie să se activeze etapa 1.

7.1.4 Sistemul de control al centralei termice

Sistemul de control al centralei termice presupune integrarea celor două funcții prezentate în secțiunile anterioare. Se va considera că funcția de încălzire apă are prioritate față de funcția de încălzire ambientală. De asemenea, sistemul de control trebuie să permită și:

- setarea parametrului de TEMPERATURĂ REFERINȚĂ APĂ DIN CIRCUIT folosind butoanele și indicatoarele din panoul de comandă;
- setarea temperaturii ambientale de referință folosind butoanele și indicatoarele din panoul de comandă;
- măsurarea temperaturii ambientale folosind senzorul STA;
- activarea funcției de încălzire ambientală dacă temperatura măsurată de STA este mai mică față de temperatura ambientală de referință, respectiv dezactivarea acesteia în caz contrar.

7.1.5 Cerințe:

1. Să se proiecteze folosind o rețea Petri logica de control a automatizării.

2. Să se testeze rețeaua Petri folosind unealta PNT00L.

Bibliografie

- [1] ***. (2012) Learning About Petri Net Toolbox. [Online]. Available: <http://www.ac.tuiasi.ro/pntool/>.
- [2] O. Păstrăvanu, M. Matcovschi, and O. Mahulea, *Aplicații ale rețelelor Petri în studierea sistemelor cu evenimente discrete*. Ed. Gh. Asachi, 2002.
- [3] O. Păstrăvanu, *Sisteme cu evenimente discrete. Tehnici calitative bazate pe formalismul rețelelor Petri*. Matrix Rom, 1997.
- [4] T. Murata, “Petri nets: Properties, analysis and applications,” *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, apr 1989.
- [5] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. Springer, 2008.