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 reacţia chimică şi de un motor (M), care prin intermediul unei elici agită conținutul rezervorului. Servovanele Y1 şi Y2 sunt folosite pentru a alimenta, respectiv goli, rezervorul.

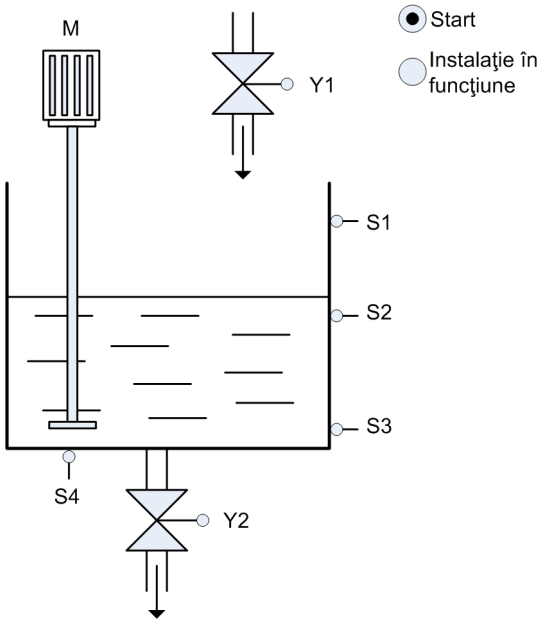

Fig. 1.1 Structura reactorului chimic

Pentru a realiza o doza de substanță chimica trebuie ca un muncitor să așeze manual

o cutie de reactiv în rezervor şi 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 sevovanei 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 "Instalaţie în funcțiune".

1.2 Cerinte

1. Să se identifice stările şi 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 şi analiza reţelelor Petri

PNTOOL (P*etri* N*et* Tool*box*) [1] este un instrument software cu ajutorul căruia se poate realiza simularea şi analiza reţelelor Petri. O descriere mult mai detaliată a modului de lucru cu PNTOOL se poate găsi în [2].

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

```
1 >>pntool
```

Listing 2.1 Pornirea interfeţei 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 reţelei Petri. În cadrul laboratorul vor fi utilizate doar reţele Petri netemporizate.

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

1) *Edit Objects*: permite moficarea obiectelor (poziţii, arce, tranziţii);
2) *Add Place*: relizează adaugarea unei poziţii la model;
3) *Add Transition*: realizează adăugarea unei tranziţii la model;
4) *Add Arc*: realizează adăugarea unui arc (de la poziţie la tranziţie sau de la tranziţie la poziţie) la model;
5) permite adăugarea de jetoane la poziţiile reţelei Petri.

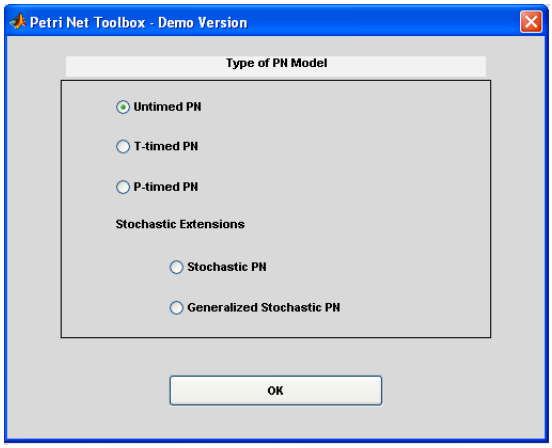

Fig. 2.1 Fereastră de selectare a modelului reţelei Petri.

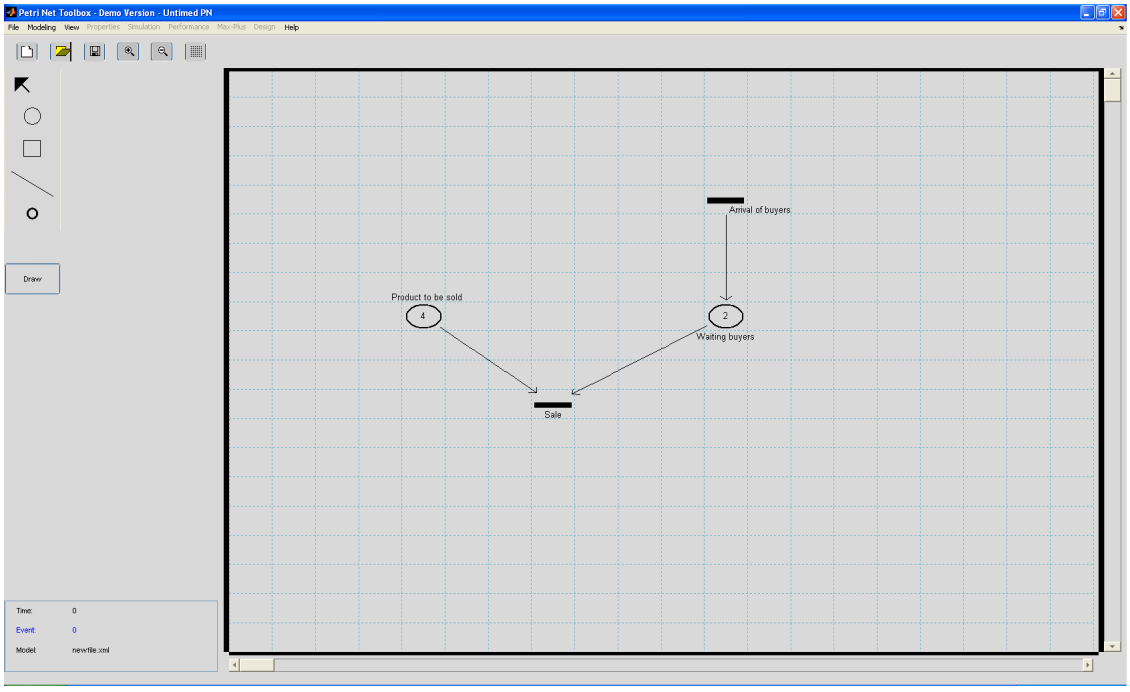

Fig. 2.2 Fereastră de desenare a reţelei Petri.

Pentru introducerea unei poziţii sau a unei tranziţii se apasă butonul *Add Place*, respectiv *Add Transition*, iar după aceea se face *click stânga* pe suprafaţa de lucru în locul unde se doreşte plasarea simbolului grafic. În continuare, pentru mutarea sau modificarea proprietăţilor poziţiei, respectiv tranziţiei, se va utiliza meniul contextual care apare după ce se face *click dreapta* pe simbolul grafic respectiv. În fereastra de modificarea a proprietăţilor unei poziţii se pot modifica, de fapt, numele poziţiei, culoarea, numărul de jetoane şi capacitatea, iar în cazul unei tranziţii se pot modifica: numele tranziţiei, mesajul care apare la executarea acesteia şi culoarea.

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

2.1.1 Setarea priorităţilor şi/sau probabilităţilor pentru tranziţiile conflictuale

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

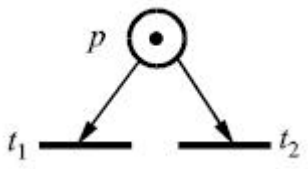

Fig. 2.3 Reţea Petri care modelează conflictul dintre evenimentele e_1 şi e_2 materializate prin tranziţiile t_1 şi respectiv t_2.

În cazul tranziţiilor conflictuale, simulatorul PNTOOL permite setarea unor priorităţi sau a unor probabilităţi de executare a tranziţiilor. Fereastra de rezolvare a conflictului dintre tranziţii 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 tranziţiilor pentru care se doreşte rezolvarea conflictului (separate prin ”,”);
- *Values*: reprezintă probabilitatea sau prioritatea de producere a fiecărei tranziţii din câmpul *Transitions*;
- *Type*: selectează modul de rezolvare a conflictului: prioritate sau probalilitate.

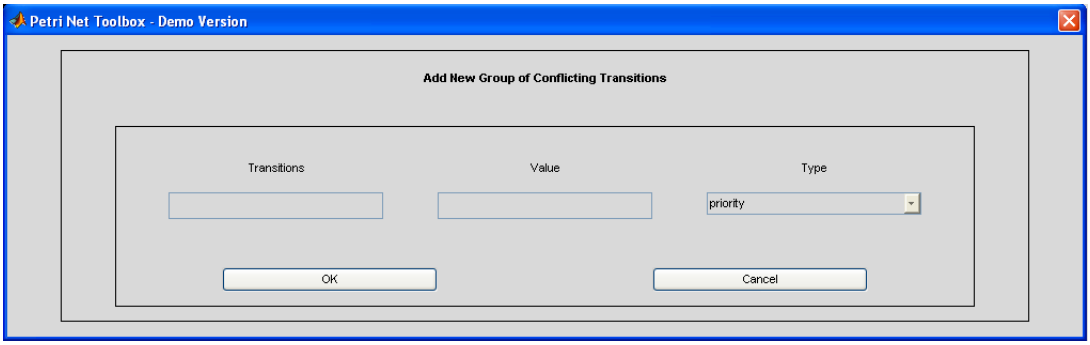

Fig. 2.4 Fereastră de stabilire a modului de rezolvare a conflictului dintre tranziţii.

Dacă modul de rezolvare a conflictului utilizează probabilităţi atunci câmpul *Values* va conţine k numere subunitare, separate prin ",", a căror sumă este 1. Fiecare număr subunitar reprezintă de fapt probabilitatea de producere a fiecărei tranziţii din câmpul *Transitions* iar k reprezintă numărul de tranziţii trecute în câmpul *Transitions*. Pentru situaţia în care se utilizează priorităţi pentru rezolvarea conflictului dintre tranziţii, atunci câmpul *Values* va conţine k numere întregi pozitive separate prin ",". Aceste numere reprezintă prioritatea tranziţiilor din câmpul *Transitions*. Se consideră că o tranziţie este prioritară faţă 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 şi în figura 2.5, panoul de simulare conţine următoarele comenzi:

- *Reset*: resetarea simulării (reţeaua Petri primeste marcajul iniţial);
- *Step*: realizează un pas de simulare (pe interfaţa grafică se poate observa tranziţia care se execută şi modificarea marcajelor);
- *Run Slow*: rulare lentă (permite vizualizarea tranziţiilor care se executa şi modificarea marcajelor);
- *Run Fast*: rulare rapidă (nu permite vizualizarea tranziţiilor care se executa şi modificarea marcajelor).

2.2 Analiza proprietăţilor comportamentale ale reţelelor Petri cu ajutorul mediului software PNTOOL

Principalele proprietăţi comportamentale ale unei reţele Petri sunt: *accesibilitatea*, *mărginirea*, *viabilitatea* şi *reversibilitatea*. Acestea proprietăţi comportamentale se pot determina dacă se analizează arborele de acoperire sau matricea de incidenţă şi ecuaţia de stare [4].

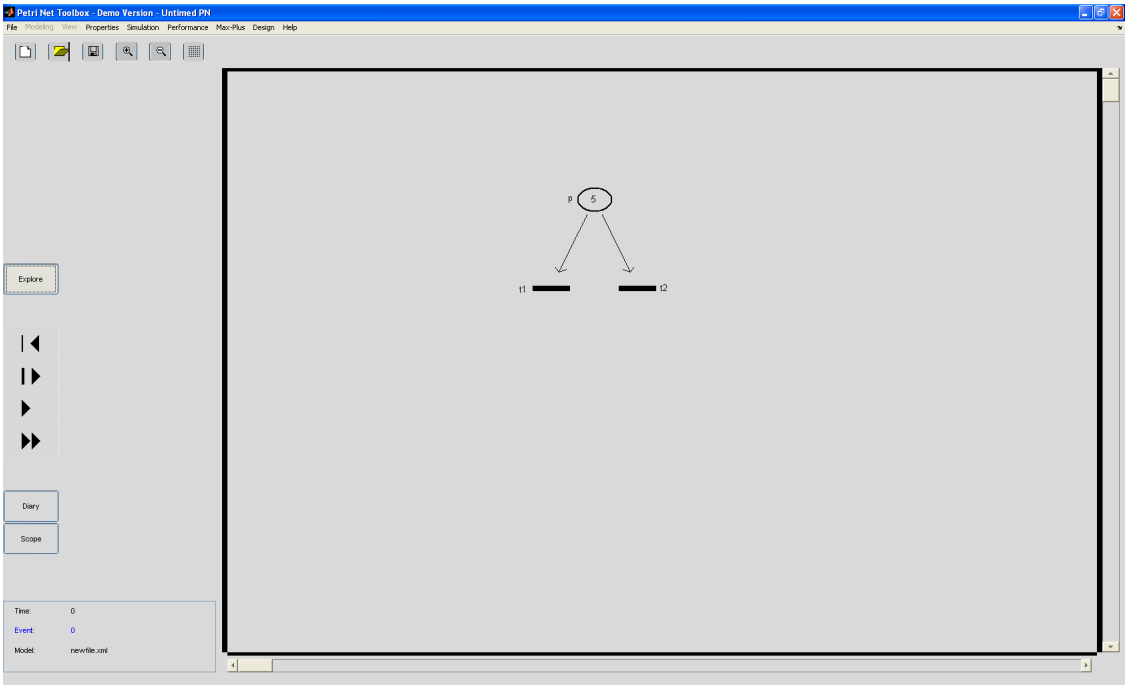

Fig. 2.5 Panoul de simulare.

2.2.1 Principalele proprietăţi comportamentale ale unei reţele Petri

2.2.1.1 Accesibilitatea

Despre un marcaj M_n se poate spune că este accesibil din marcajul iniţial M_0 dacă există o secvenţă de executări de tranziţii care transformă M_0 în M_n [2]. Secvenţa de executări de tranziţii se notează prin:

$$\sigma = M_0 t_1 M_1 t_2 M_2 ... t_n M_n, \tag{2.1}$$

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

$$\sigma = t_1 t_2 ... t_n. \tag{2.2}$$

2.2.1.2 Mărginirea

O reţea Petri cu capacitate infinită este mărginită dacă numărul de jetoane din fiecare poziţie nu depăşeşte un număr finit k pentru orice marcaj accesibil din marcajul iniţial M_0. În limbaj matematic, o reţea Petri cu capacitate infinită este mărginită dacă:

$$M(p) \leq k \;\; \text{pentru } \forall p \in P \text{ şi } \forall M \in R(M_0), \tag{2.3}$$

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

2.2.1.3 Viabilitatea

O reţea Petri este viabilă dacă, indiferent de marcajul care a fost atins pornind din M_0, este posibil ca, în continuare să fie executată orice tranziţie t a reţelei. Până la executarea tranziţiei t se poate executa un număr finit de alte tranziţii [2].

Un marcaj pentru care nicio tranziţie a reţelei nu mai poate fi executată se numeşte *marcaj de deadlock*.

2.2.2 Arborele de acoperire

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

1) se stabileşte M_0 ca rădăcină şi se etichetează ca *marcaj nou*;
2) pentru fiecare marcaj M etichetat drept *marcaj nou*, se efectuează unul dintre următorii subpaşi:
   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 tranziţie validată, atunci M se etichetează ca *marcaj de deadlock*;
   2.3) dacă pentru M există tranziţii validate, atunci pentru fiecare tranziţie validată t se efectuează următoarele etape:
      – se obţine marcajul M' care rezultă din executarea tranziţiei 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 poziţie p şi $M' \neq M$" atunci $M'(p)$ se înlocuieşte cu ω pentru fiecare poziţie 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 tranziţiei t şi se etichetează M' drept *marcaj nou*.

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

2.2.3 Matricea de incidenţă şi ecuaţia de stare

Se numeşte *matrice de incidenţă* a unei reţele Petri o matrice de dimensiune $n \times m$, unde n reprezintă numărul tranziţiilor, iar m numărul poziţiilor. Matricea de incidenţă este de forma $\boldsymbol{A} = [a_{ij}]$ ale cărei elemente sunt numere întregi:

$$a_{ij} = a_{ij}^{+} - a_{ij}^{-},\ i = 1, ..., n,\ j = 1, ..., m, \tag{2.4}$$

unde:

– $a_{ij}^{+} = W(t_i, p_j)$ reprezintă ponderea arcului de la tranziţia t_i, către poziţia p_j;

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

Matricea de incidenţă a unei reţele Petri se obţine cu ajutorul mediului software PN-TOOL, în urma transmiterii comenzii: *Properties -> Incidence Matrix*.

Cunoscând matricea de incidenţă, se poate obţine matricea de stare a reţelei Petri:

$$M_k = M_{k-1} + \boldsymbol{A}^{\mathrm{T}} \boldsymbol{u_k}, \tag{2.5}$$

unde, M_k este marcajul care se obţine din marcajul M_{k-1} dacă se execută tranziţia t_i. Vectorul coloană $\boldsymbol{u_k}$, de dimensiune $n \times 1$, are toate elementele 0 cu excepţia celui de al i-lea element care este 1 (corespunzător tranziţie t_i).

Marcajul în care se ajunge după executatea unei secvenţe de q tranziţii (pornind din marcajul iniţial) se poate obţine cu ajutorul relaţiei:

$$M_q = M_0 + \boldsymbol{A}^{\mathrm{T}} \sum_{k=1}^{q} \boldsymbol{u_k}, \tag{2.6}$$

2.2.4 Teme aplicative

2.2.4.1 Arbore de accesibilitate şi matrice de incidenţă

Să se construiască arborele de accesibilitate şi matricea de incidenţă pentru reţelele Petri din figura 2.6.

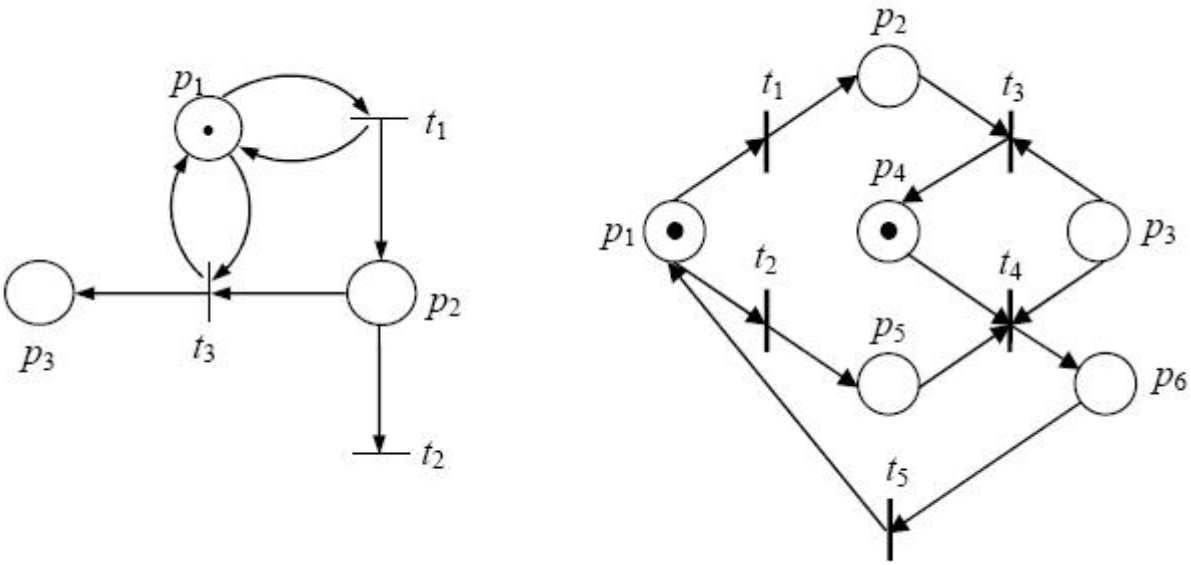

Fig. 2.6 Reţele Petri.

2.2.4.2 Sistem de calcul

Se consideră un sistem de calcul biprocesor, echipat cu două unităţi de disc D_1, D_2. Fiecare dintre cele două procesoare P_1 şi 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 şi 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 şi D_2 sunt eliberate simultan. O reprezentare schematizată a funcţionării sistemului de calcul este dată în figura 2.7. La momentul iniţial atât procesoarele cât şi discurile sunt libere.

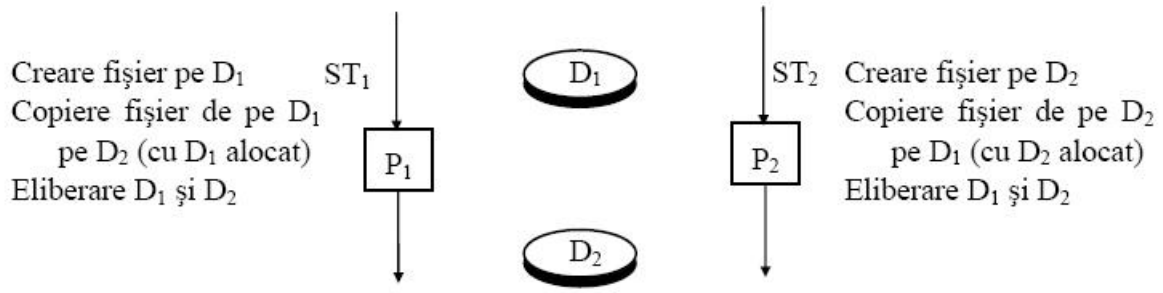

Fig. 2.7 Structură sistem calcul.

Cerinţe:

1) să se construiască un model de tip reţea Petri pentru funcţionarea sistemului de calcul, precizându-se semnificaţia fizică a poziţiilor şi tranziţiilor;
2) să se construiască arborele de accesibilitate şi matricea de incidenţă pentru modelul obţinut la punctul anterior;
3) să se investigheze posibilitatea producerii fenomenului de deadlock în sistem; se va preciza marcajul reţelei Petri pentru care se ajunge în deadlock precum şi secvenţele de executări de tranziţii 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 reţelelor Petri

PetriNetExec este o bibliotecă Java care permite integrarea reţelelor Petri în aplicaţiile Java. Cu ajutorul bibliotecii PetriNetExec în aplicaţia Java se pot introduce poziţii şi tranziţii; se pot lega acestea cu arce; se poate defini marcajul iniţial; se pot trimite evenimente şi se poate vedea fluxul de jetoane din reţea.

Pentru o înţelegere mai uşoară se va prezenta un exemplu de implementare a unei reţele Petri în PetriNetExec.

În figura 3.1 este prezentat modelul cu reţea Petri a comportamentului unui vehicul. Iniţial, agentul este oprit (jetonul se afla în p_1) şi trece în starea de accelerare (jetonul se află în p_2) dacă se produce evenimentul asociat tranziţiei t_1 (s-a apăsat pedala de acceleraţie). În continuare, vehiculul poate să încetinească (jetonul se află în p_3), dacă se produce evenimentul asociat tranziţiei t_3 (s-a apăsat pedala de frână), sau poate să menţină viteza constantă (jetonul se află în p_4) dacă se produce evenimentul asociat tranziţiei t_7 (vehiculul a atins viteza dorită).

Pentru început se crează o reţea 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 reţelei Petri

În continuare, se adaugă poziţiile reţelei Petri, conform listing-ului 3.2. Primul parametru reprezintă identificatorul poziţiei, cel de-al doilea parametru este un şir de caractere care poate fi utilizat la depanare, iar cel de-al treilea parametru reprezintă numărul de jetoane care se află în poziţie la momentul iniţial.

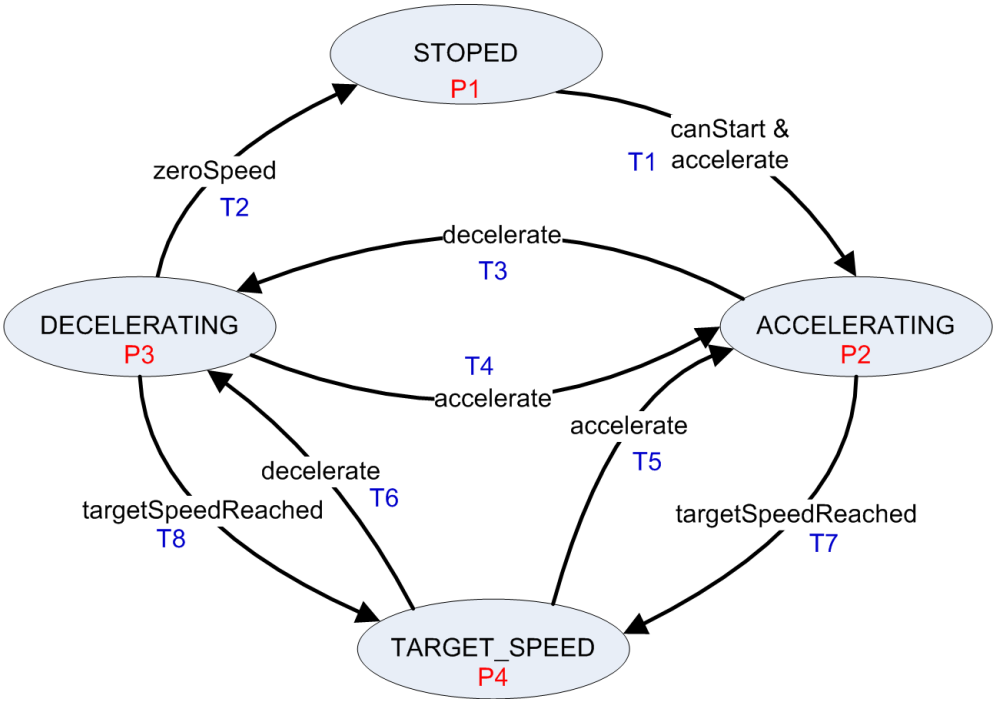

Fig. 3.1 Modelul cu reţea Petri al comportamentului unui vehicul.

```
String p1_ID = "P1";
String p2_ID = "P2";
String p3_ID = "P3";
String p4_ID = "P4";

net.addPlace(p1_ID, "STOPED", 1);
net.addPlace(p2_ID, "ACCELERATING", 0);
net.addPlace(p3_ID, "DECELERATING", 0);
net.addPlace(p4_ID, "TARGET_SPEED", 0);
```

Listing 3.2 Adăugarea poziţiilor la reţeaua Petri.

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

```
String t1_ID = "T1";
String t2_ID = "T2";
String t3_ID = "T3";
String t4_ID = "T4";
String t5_ID = "T5";
String t6_ID = "T6";
String t7_ID = "T7";
String t8_ID = "T8";

net.addTransition(t1_ID, "canStart_and_accelarate", Transition.TransitionType.
    NORMAL);
```

```
net.addTransition(t2_ID, "zeroSpeed", Transition.TransitionType.NORMAL);
net.addTransition(t3_ID, "decelerate", Transition.TransitionType.NORMAL);
net.addTransition(t4_ID, "accelarate", Transition.TransitionType.NORMAL);
net.addTransition(t5_ID, "accelarate", Transition.TransitionType.NORMAL);
net.addTransition(t6_ID, "decelerate", Transition.TransitionType.NORMAL);
net.addTransition(t7_ID, "targetSpeedReached", Transition.TransitionType.
    NORMAL);
net.addTransition(t8_ID, "targetSpeedReached", Transition.TransitionType.
    NORMAL);
```

Listing 3.3 Adăugarea tranziţiilor la reţeaua Petri.

Conectarea unui arc de la o poziţie la o tranziţie se face cu ajutorul metodei *connectToOutputTransition* (vezi listing 3.4) care conţine următorii parametrii:

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

```
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);
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);
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);
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 poziţii la tranziţii.

Conectarea unui arc de la o tranziţie la o poziţie se face cu ajutorul metodei *connectToOutputPlace* (vezi listing 3.5) care conţine următorii parametrii:

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

```
net.retrieveTransition(t1_ID).connectToOutputPlace(net.retrievePlace(p2_ID),
    null, "t1 to p2", 1);
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);
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);
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);
net.retrieveTransition(t8_ID).connectToOutputPlace(net.retrievePlace(p4_ID),
    null, "t8 to p4", 1);
```

Listing 3.5 Conectarea arcelor de la tranziţii la poziţii.

În continuare, pentru verificare, se pot afişa poziţiile şi tranziţiile utilizând metodele statice ale clasei *PetriNetUtils* (vezi listing 3.6).

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

Listing 3.6 Afişarea poziţiilor şi tranziţiilor din reţeaua Petri.

După construcţia reţelei Petri trebuie realizat un generator de evenimente, cu ajutorul căruia se va efectua executarea tranziţiilor validate. De asemenea, generatorul trebuie anunţat ce reţea Petri îi ascultă evenimentele (vezi listing 3.7).

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

Listing 3.7 Adăugarea unui generator de evenimente la reţeaua Petri

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

```
net.retrievePlace(p1_ID).createExitCallback(1, new AgentPlaceAction(this.
    context){
        @Override
        public void callback(PetriNet net, Place place)
        {
            this.getAgentContext().agent.actionStopedExit();
        }
    });

....

public void actionStopedExit(){
    System.out.println("STOPED exit");
```

```
}
```

Listing 3.8 Adăugarea unui mecanism de callback pentru poziţia p_1.

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

– *createEntryCallback*: când poziţia primeşte un jeton;

– *createExitCallback*: când marcajul poziţiei scade sub o anumită valuare;

– *createEntryCallback*: când marcajul poziţiei creşte peste o anumită valuare.

Pentru tranziţii se pot defini următoarele callback-uri:

– *createEntryCallback*: când se execută tranziţia.

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

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

Listing 3.9 Verificarea reţelei Petri.

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

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

Listing 3.10 Transmisie de evenimente către reţeaua Petri.

4. Panoul de comandă al unei automatizării

4.1 Caietul de sarcini

In figura 4.1 este prezentă schema unui panou de comandă. Panoul conţine un LCD care are patru zone în care se pot afişa mesaje text. Fiecare zonă poate să conţină un mesaj text de 6 caractere. De asemenea, caracterul "*" poate să preceadă textele din cele 4 zone. În acest fel se alege care funcţie (mesaj text) este selectată. Butoanele *Sus* şi *Jos* sunt folosite pentru a muta caracterul "*" dintr-o zonă în alta. Butonul *Select* este folosit pentru a selecta funcţia dorită.

Atomatizarea mai conţine un indicator (led) şi un servo mecanism a căror stare se modifică pe baza funcțiilor selectate pe LCD.

4.1.1 Modul de funcționare al panoului de comandă

În momentul în care automatizare pornește următoarele texte trebuie să fie afișate pe LCD:

- Zona 1: Ind
- Zona 2: Servo
- Zona 3: [nefolosit]
- Zona 4: [nefolosit]

De asemenea, caracterul "*" trebuie să se găsească iniţial în zona 1. Mai departe, prin apăsarea butoanelor *Sus*, *Jos* se va muta corespunzător caracterul "*" dintr-o zonă în alta.

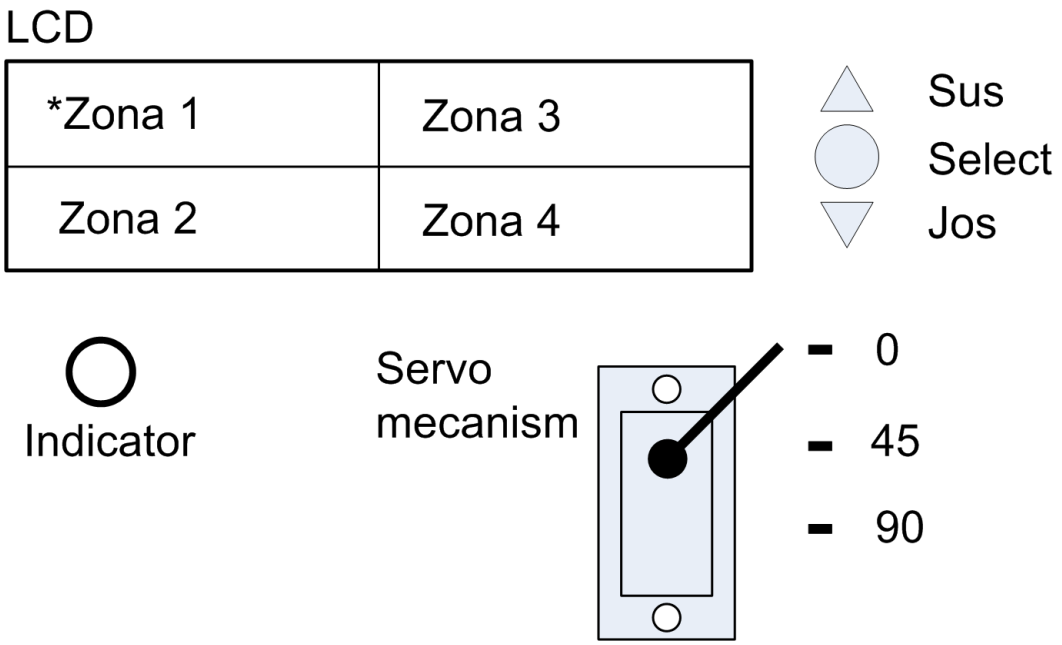

Fig. 4.1 Schema panoului de comandă.

Dacă se apasă butonul *Select* şi caracterul "*" se află în dreptul textului Ind (zona 1), pe LCD trebuie să se afişeze următoarele texte:

- Zona 1: Oprit
- Zona 2: Pornit
- Zona 3: Inapoi
- Zona 4: [nefolosit]

Iar caracterul "*" trebuie să se găsească în zona 1. Butoanele *Sus* şi *Jos* au o funcționare similară ca în cazul anterior. În momentul în care este selectată zona 2 (textul Pornit) trebuie ca indicatorul să fie activat. Butonul *Select* poate este activ doar în zona 3 şi va conduce la revenirea la meniul iniţial.

Dacă se apasă butonul *Select* şi caracterul "*" se află în dreptul textului Servo (zona 2), pe LCD trebuie să se afişeze următoarele texte:

- Zona 1: 0
- Zona 2: 45
- Zona 3: 90
- Zona 4: Inapoi

Iar caracterul "*" trebuie să se găsească în zona 1. Butoanele *Sus* şi *Jos* au o funcționare similară ca în cazul anterior. Butonul *Select* poate este activ doar în zona 4 şi va conduce la revenirea la meniul iniţial. Servo mecanismul trebuie să-şi schimbe starea în funcţie de valoare zonei selectate.

4.1.2 Cerinţe:

1. Să se proiecteze folosind o reţea Petri logica de control a automatizării.
2. Să se testeze reţeaua Petri folosind unealta PNTOOL.

5. Sistem de fabricaţie cu trei puncte de lucru.

În figura 5.1 este prezentat un sistem de fabricaţie alcătuit din trei puncte de lucru, iar în tabelul 5.1 se află descrierea mărimilor de intrare şi ieşire.

Tab. 5.1 Descrierea mărimilor de intrare şi ieşire ale sistemului de fabricaţie 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 extremitaţi piston P_1
S_3, S_4	senzori extremitaţi piston P_2
S_5, S_6	senzori extremitaţi piston P_3
S_7, S_8	senzori extremitaţi piston P_4
S_9, S_{10}	senzori extremitaţi piston P_5
S_{11}	senzor prezenţă piesă la punctul de presare
S_{12}	senzor de temperatură
S_{13}	senzor prezenţă piesă la punctul de pulverizare
S_{14}	senzor trecere piese către punctul de activare adeziv
AL	alarmă magazie de piese goală

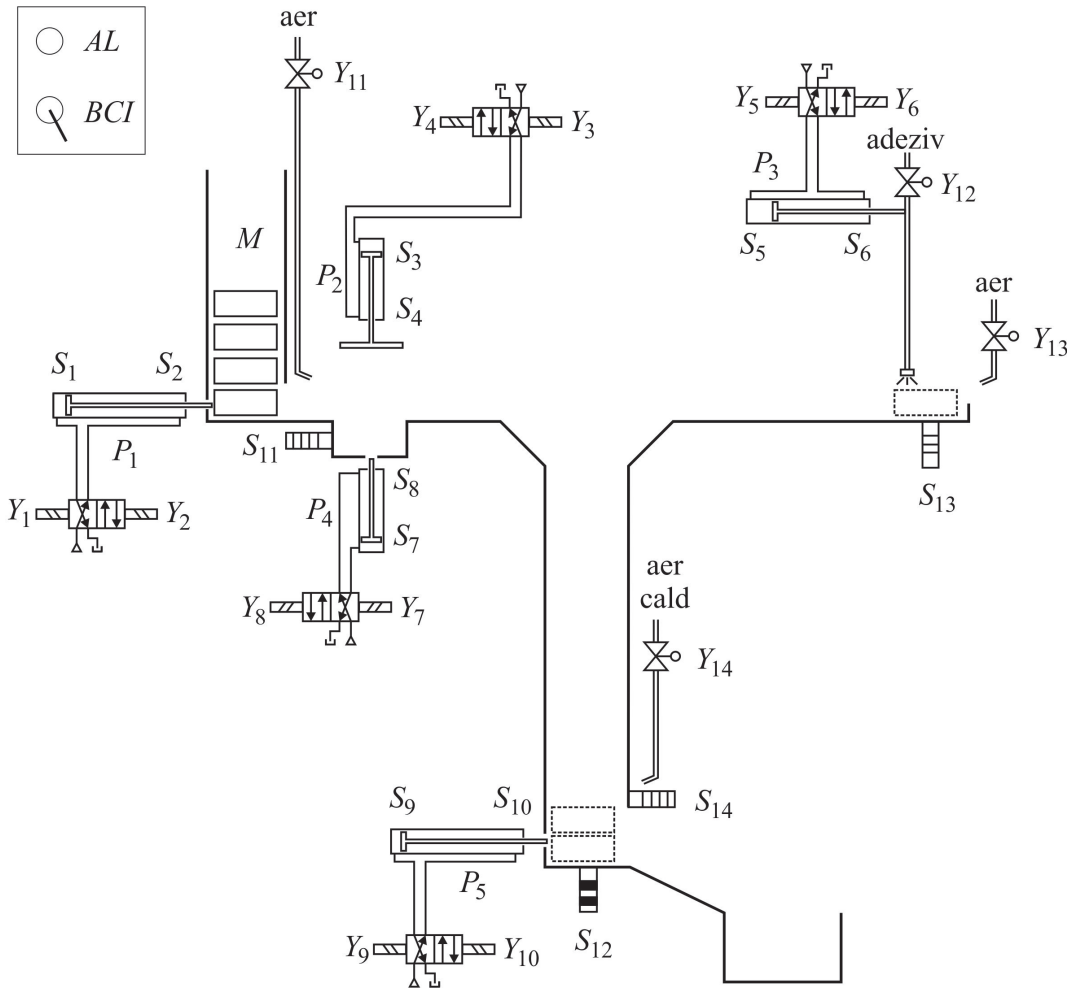

Fig. 5.1 Schema generală a sistemului de fabricaţie cu trei puncte de lucru.

Primul punct de lucru realizează pulverizarea cu adeziv a unei piese. Această operaţie se realizează prin transmiterea comenzii Y_6 care conduce la avansul pistonului P_3. În acelaşi timp trebuie pornit şi pulverizatorul de adeziv cu ajutorul comenzii Y_{12}. După finalizarea procesului de pulverizare adeziv (a fost activat senzorul S_6) se opreşte pulverizatorul prin dezactivarea comenzii Y_{12} şi 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 obţine în urma activării comenzii Y_{13}.

Cel de-al doilea punct de lucru realizează ştanţarea 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 şi situaţia 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 conţinea cel puţin o piesă. Dacă nu se generează acest eveniment şi pistonul P_1 ajunge la capătul de cursă (s-a produs evenimentul S_2) înseamnă că magazia nu conţinea piese. În această situaţie se aprinde indicatorul de alarmă AL şi se aşteaptă 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 locaşul pentru ştanţare, se transmite comanda Y_4 pentru ca pistonul P_2 să înceapă coborârea. Se va considera că piesa este ştanţată în momentul în care s-a generat evenimentul asociat senzorului S_4. În continuare se aşteaptă retragerea pistonului P_2 (după ce s-a transmis comanda Y_4). Pentru a descărca piesa din punctul de ştanţare se acţionează pistonul P_4 prin transmiterea comenzii Y_8 şi în acelaşi timp se porneşte jetul de aer cu ajutorul vanei comandate de mărimea de ieşire 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 primeşte comanda Y_{10} până când piesa este evacuată (s-a generat evenimentul S_{10}), după care primeşte comanda Y_9 până când se retrage pistonul.

Modul de operare al instalaţiei este următorul:

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

5.0.1 Cerinţe:

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

2. Să se testeze reţeaua Petri folosind unealta PNTOOL.

6. Sistem de fabricaţie cu resurse partajate.

În figura 6.1 este prezentat un sistem de fabricaţie cu resurse partajate, iar în tabelul 6.1 se află descrierea mărimilor de intrare şi ieşire.

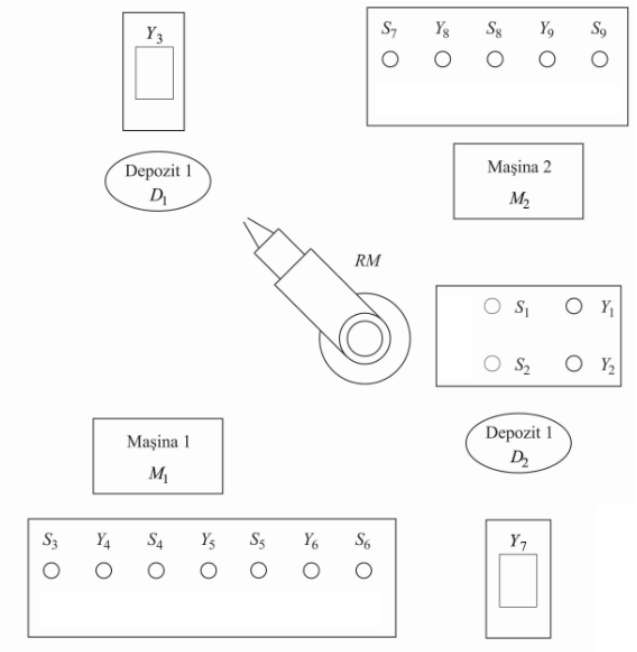

Fig. 6.1 Schema generală a sistemului de fabricaţie cu resurse partajate.

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

Tab. 6.1 Descrierea mărimilor de intrare şi ieşire ale sistemului de fabricaţie cu resurse partajate.

Mărime	Descriere
Y_1	RM încarcă o piesă pe maşina M_1
Y_2	RM încarcă o piesă pe maşina M_2
Y_3	numărul de paleţi din depozitul D_1
Y_4	maşina M_1 prelucrează o piesă
Y_5	maşina M_1 aşteaptă descărcarea
Y_6	maşina M_1 descarcă piesa prelucrată
Y_7	numărul de paleţi din depozitul D_2
Y_8	maşina M_2 prelucrează o piesă
Y_9	maşina M_2 descarcă piesa prelucrată
S_1	comanda RM pentru încărcarea unei piese pe maşina M_1
S_2	comanda RM pentru încărcarea unei piese pe maşina M_2
S_3	finalizarea încărcării unei piese pe maşina M_1
S_4	finalizarea prelucrării unei piese pe maşina M_1
S_5	comanda descărcării piesei prelucrate pe maşina M_1
S_6	finalizarea descărcării piesei prelucrate pe maşina M_1
S_7	finalizarea încărcării unei piese pe maşina M_2
S_8	finalizarea prelucrării unei piese pe maşina M_2
S_6	finalizarea descărcării piesei prelucrate pe maşina 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 maşina unealtă M_1. Pe durata mutării trebuie ca indicatorul Y_1 să fie activat. În mod similar, senzorul S_2 şi indicatorul Y_2 sunt folosiţi pentru a realiza mutarea unei piese din depozitul D_2 pe maşina unealtă M_2.

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

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

– în depozitul D_1 se găseşte cel putin o piesă;

– maşina unealtă M_1 este disponibilă (a descărcat ultima piesă prelucrata);

– 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 maşina unealtă M_1. Maşina 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, maşina unealtă aşteaptă 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. Maşina unealtă M_1 va realiza descărcarea doar dacă în depozitul D_2 sunt mai puţin 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 maşina unealtă M_2 doar dacă următoarele condiţii sunt îndeplinite simultan:

- în depozitul D_2 se găseşte cel putin o piesă;
- maşina unealtă M_2 este disponibilă (a descărcat ultima piesă prelucrata);
- robotul mobil este disponibil;
- se apasă butonul S_2.

Senzorul S_7 sesizează momentul în care o piesa este încărcată pe maşina unealtă M_2. Mai departe aceasta prelucrează piesa şi activează indicatorul Y_8. Finalizarea prelucrării este sesizată de senzorul S_8. După finalizarea unei piese, maşina 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 maşina unealtă descarcă o piesă, paletul este automat reîncărcat cu o piesă brută şi stocat în depozitul D_1.

6.0.1 Cerinţe:

1. Să se proiecteze folosind o reţea Petri logica de control a automatizării.
2. Să se testeze reţeaua Petri folosind unealta PNTOOL.

7. Controlul supervizat al roboţilor autonomi dintr-o incintă.

În figura 7.1 este prezentată schema incintei unde operează roboţii autonomi, iar în tabelul 7.1 se află descrierea mărimilor de intrare şi ieşire.

Tab. 7.1 Descrierea mărimilor de intrare şi ieşire ale sistemului de control supervizat al roboţilor.

Mărime	Descriere
$Garaj$	numărul de roboţi din spaţiul de garare
C_1	numărul de roboţi din camera 1
C_2	numărul de roboţi din camera 2
C_3	numărul de roboţi 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 iniţial în garaj se găsesc 6 roboţi. Utilizatorul uman poate să comande roboţii să se deplaseze dintro camera în alta. Comanda de mutare se realizează prin apăsarea unuia dintre butoanele B_1 ... B_6 (vezi tabelul 7.1).

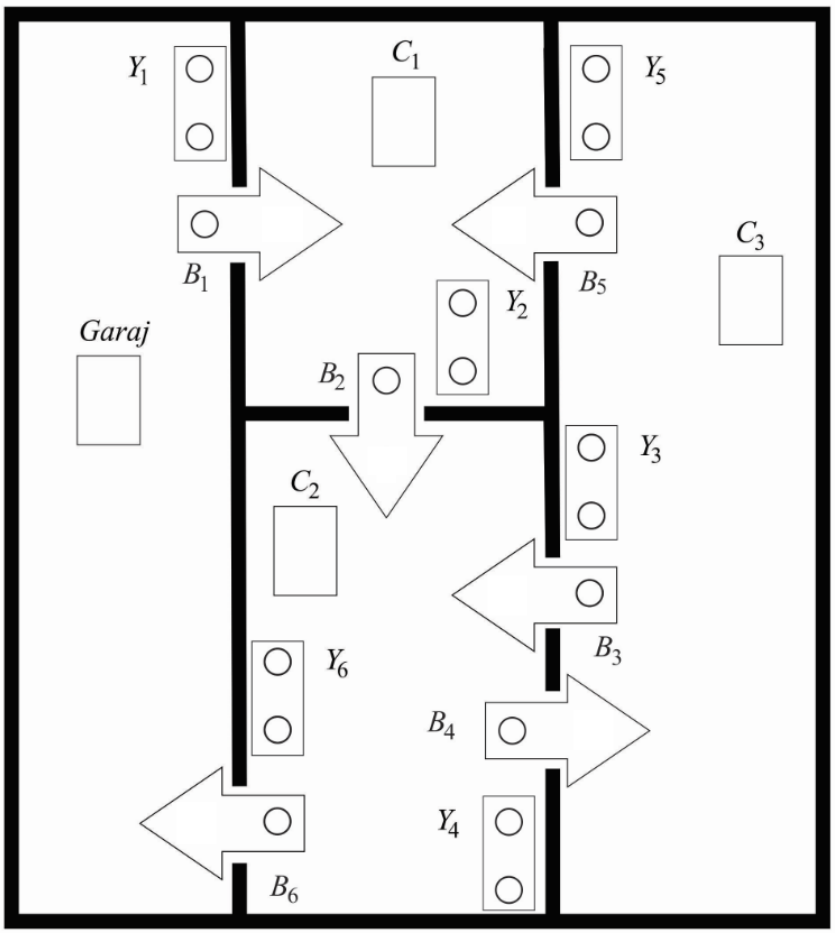

Fig. 7.1 Schema incintei unde operează roboţii autonomi.

Incinta unde operează roboţii are următoarele constrângeri:

– în camera 1 pot intra maxim 2 roboţi;

– numărul total de roboţi din camerele 1 şi 2 nu poate să fie mai mare de 4;

– numărul total de roboţi din camerele 2 şi 3 nu poate să fie mai mare de 3.

Indicatoarele $Garaj$, C_1, C_2 şi C_3 sunt folosite pentru a afişa numărul de roboţi din camera respectivă. Indicatoarele Y_1 ... Y_6 sunt activate în momentul în care nu se poate intra în camera respectivă.

7.0.1 Cerinţe:

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

2. Să se testeze reţeaua Petri folosind unealta PNTOOL.

Bibliografie

[1] ***. (2012) Learning About Petri Net Toolbox. [Online]. Available: http://www.ac.tuiasi.ro/pntool/.

[2] O. Păstrăvanu, M. Matcovschi, and O. Mahulea, *Aplicaţii ale reţelelor Petri în studierea sistemelor cu evenimente discrete.* Ed. Gh. Asachi, 2002.

[3] O. Păstrăvanu, *Sisteme cu evenimente discrete. Tehnici calitative bazate pe formalismul reţelelor 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.