UNIVERSITATEA "POLITEHNICA" TIMIŞOARA
FACULTATEA DE MECANICĂ

Ing. MIRCEA FLORINEL DREUCEAN

OPTIMIZAREA FLUXULUI DE MATERIALE ŞI INFORMAŢII ÎN SISTEME DE FABRICAŢIE FLEXIBILĂ ROBOTIZATE

TEZA DE DOCTORAT

Conducător ştiinţific:

Profesor doctor inginer FRANCISC KOVACS

MAI
1999

CUVÂNT ÎNAINTE

Domeniul sistemelor flexibile de fabricație este fascinant deopotrivă pentru specialişti cât şi pentru neavizați, prin îngemănarea sub diverse forme a unor tehnici foarte diferite: tehnica maşinilor de lucru de mare performanță, tehnica sculelor şi dispozitivelor inteligente, tehnicile de conducere bazate pe inteligență artificială, tehnicile de proiectare constructivă şi tehnologică bazate pe sisteme software de mare performanță, tehnici de conducere automată cu calculatoare deosebit de puternice şi, nu în ultimul rând, tehnicile de redefinire a locului operatorului uman în astfel de medii industriale de un pragmatism extrem. Toate aceste aspecte de mare performanță ale sistemelor de fabricație, alături de conceptul însuşi de fabricație flexibilă, s-au născut din cerințele impuse producătorilor de bunuri de orice fel de către competiția acută care se derulează la nivelul pieței economice a lumii care se dezvoltă din ce în ce mai mult spre o piață unică.

Cercetătorii români în domeniu au descoperit în ultimii ani noile concepte ale fabricației flexibilă şi au putut să vadă la lucru astfel de sisteme în diverse locuri de pe glob. Cu certitudine că, atunci când realitatea economică o va impune, şi în țara noastră se vor construi astfel de sisteme, sub competența celor care astăzi, mânați de un optimism specific spiritului academic, continuă să se pregătească şi să-şi asigure un loc printre caracterele contemporaneității.

Pentru autorul lucrării de față, şansa de a descoperi ceva din mirajul acestei lumi a tehnicii de vârf a constituit-o prima vizită de documentare făcută în Finlanda, La Institutul de

Tehnologie din Kuopio. Ceea ce până atunci părea un capitol de roman de ficțiune s-a arătat a fi o realitate de fiecare zi pentru mediul industrial finlandez, unde "SISU" nu este doar o marcă de camioane de mare tonaj, produse în fabrici ale mileniului următor, dar şi un sinonim pentru omul harnic şi hotărât să sfarme munții din cale pentru a-şi atinge un scop. Şansa de a fi acolo i-o datorez profesorului Francisc Kovacs, conducătorul ştiințific al acestei lucrări, omul de la care am învățat, printre multe alte lucruri, că etica academică impune o deschidere permanentă spre nou dacă vrei să fii prețuit şi să te impui într-un anumit domeniu. Ii mulțumesc cu acest prilej pentru încrederea pe care mi-a acordat-o atunci când, proaspăt revenit din documentarea în străinătate, i-am propus titlul tezei de doctorat pe care am legat-o între aceste coperți. Ii mulțumesc de asemenea pentru îndemnul permanent pe care mi l-a adresat de a încerca noi orizonturi, tot mai departe de pregătirea mea de bază şi totuşi mereu mai aproape prin legăturile nebănuite care mi s-au dezvăluit.

Aş dori de asemenea să adresez mulțumiri şi colectivului în care am lucrat şi care, prin calitățile pe care le-a dobândit datorită relațiilor profesionale şi ştiințifice dintre membrii lui, a devenit mult mai mult decât suma părților sale, aşa cum se afirmă de fapt şi în teoria sistemelor. Si tot pe această bază pot să susțin că, dacă această lucrare are unele calități, oricare ar fi ele, acestea se datoresc în primul rând relațiilor care m-au condus în evoluția mea ca parte a sistemului.

Doresc să adresez un cuvânt de mulțumire referenților ştiințifici ai prezentei lucrări pentru grija cu care s-au aplecat asupra textului şi sugestiile importante pe care le-au făcut în legătură cu continuarea cercetărilor în domeniu.

Autorul

CUPRINS GENERAL

1. REFLECTAREA PROBLEMATICII FLUXURILOR DE MATERIALE ŞI DE INFORMAŢIE DIN SISTEMELE FLEXIBILE DE FABRICAŢIE ÎN LITERATURA DE SPECIALITATE

1.1. Cuprins

1.2. Elemente generale de teoria sistemelor

Noțiunea de sistem este preluată din limba greacă, unde este folosită în sensul de "întreg format din părți". Definiția cea mai completă folosită astăzi pentru această noțiune este "*Un sistem este o reuniune de elemente împreună cu mulțimea relațiilor ce există între acestea*". Se deduce de aici faptul că un sistem este mai mult decât mulțimea părților sale. Înțelesul exact al definiției este că sistemul nu se obține prin simpla adiționare a părților, ci prin integrarea lor în sensul de "uniune şi influențare reciprocă a părților unui întreg". Deci ceea ce este în primul rând un sistem nu este componența lui elementară, ci relațiile ce se stabilesc între aceste elemente [Ropohl, G., 1971, pag. 110 şi urm.]. O altă definiție a unui sistem este "*ansamblul de obiecte organizate în jurul unui scop, integrate într-un mediu ambiant*" [Gallou, F. le, Buchon-Meunier, B., 1992, pag. 12]. In mod simbolic, definiția de mai sus se poate exprima prin relația următoare:

$$\{S\} = \{E; R; O; R_e\} \qquad \textbf{(1.2-1)}$$

In această relație simbolurile au următoarea semnificație:

{E}- ansamblul elementelor componente ale sistemului;

{R}- ansamblul relațiilor interne, sau structura sistemului;

{O}- ansamblul obiectivelor sau al finalității sistemului;

{Re}- ansamblul relațiilor exterioare cu mediul.

Pentru ca un ansamblu de elemente să poată constitui un sistem, el trebuie să răspundă unui *grup de cerințe*, prezentate în continuare, precum şi tuturor regulilor ce derivă din acestea. Aceste cerințe sunt:

- *integralitatea sau coerența*, care afirmă că toate elementele şi subsistemele unui sistem sunt interconectate, adică au cel puțin o relație de un anumit gen cu un alt element din sistem;
- *autonomia*, care constă în relativa independență a sistemului față de mediul exterior;
- *finalitatea*, care afirmă că existența unui sistem nu are sens decât în măsura în care evoluția lui este coerentă, orientată spre un anumit scop;
- *dinamica*, provenind din interconectarea diverselor elemente prin relații interioare care duc la modificări în timp ale caracteristicilor acestora;

- *permanența sau devenirea*, legată de evoluția în timp a sistemului care se desfăşoară între jaloane precise, asemănătoare evoluției sistemelor vii: naştere, viață şi moarte.

Din cerințele de mai sus se pot desprinde câteva *reguli* mai importante ce se manifestă în funcționarea unui sistem:

- un sistem este întotdeauna inclus într-un sistem de rang superior (cu semnificație de mediu ambiant);
- un element al sistemului nu este în mod necesar un subsistem;
- un sistem nedefinit ca obiective şi finalități nu poate fi descris în mod coerent şi univoc;
- un sistem are întotdeauna o activitate şi o evoluție, chiar dacă momentan el se află în repaos.

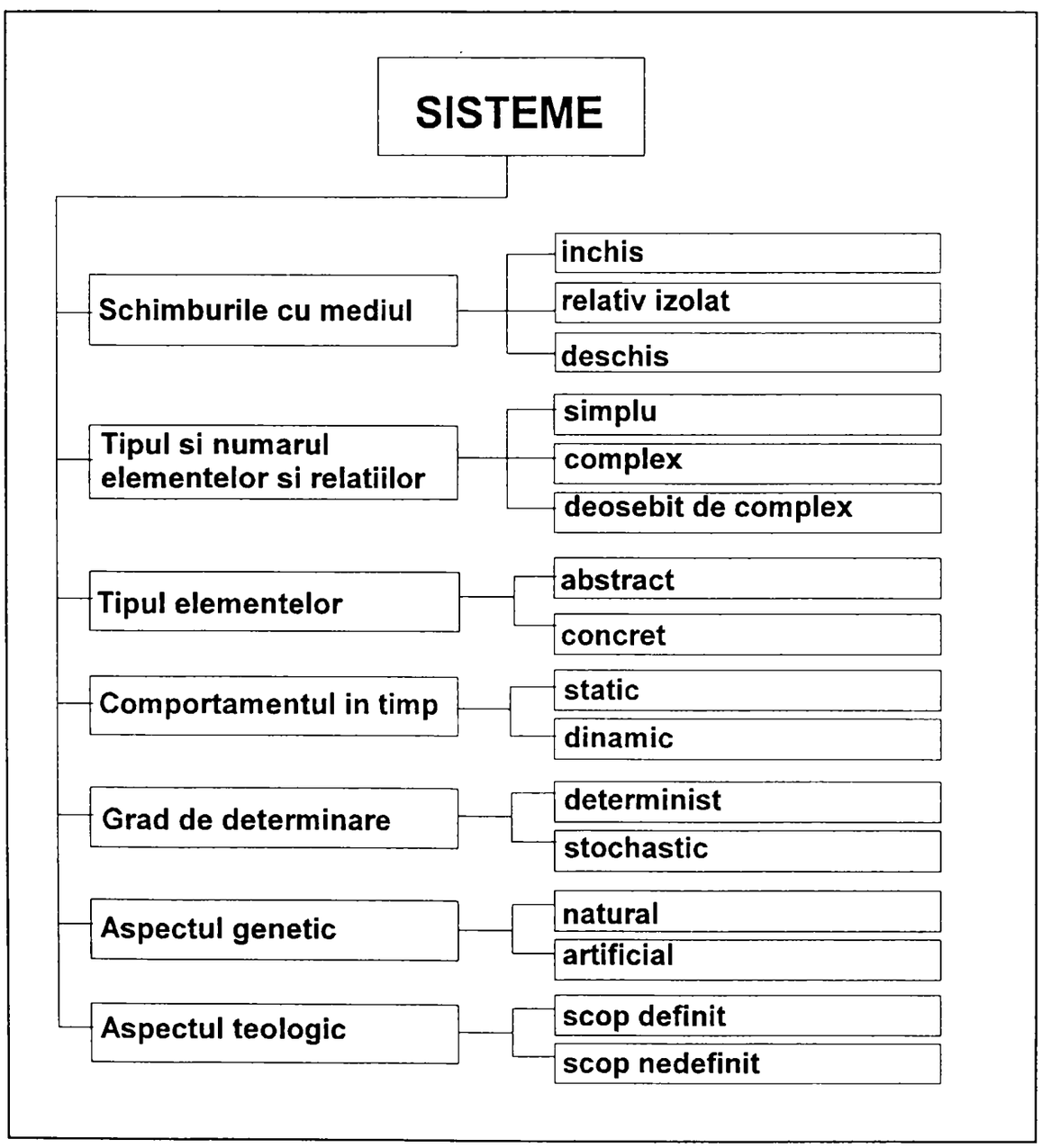


Figura 1.2-1

Clasificarea sistemelor se poate face după relaţiile pe care acestea le au cu mediul "*M*", înţelegând prin mediu mulţimea complementară "$\overline{S}$" a mulţimii elementelor sistemului "S". Acest aspect al relaţiei dintre sistem şi mediu poate fi exprimat matematic astfel:

$$S \cup \overline{S} = M$$

(1.2-2)

In diagrama din Figura 1.2-1 este prezentată schematic această clasificare.

1.3. Funcţia unui sistem

Un sistem relativ izolat "S" are intrările "x_1, x_2 ,.....,x_p", şi ieşirile "y_1, y_2, ...,y_p". Aceste două seturi de date pot fi scrise sub forma unor vectori:

$$\vec{x} = (x_1, x_2, ..., x_p)$$
$$\vec{y} = (y_1, y_2, ..., y_p)$$

(1.3-1)

Funcţia unui sistem este de a transforma setul de date de intrare în setul de date de ieşire, deci de a transforma vectorul de intrare "$\vec{x}$" în vectorul de ieşire "$\vec{y}$" [Kovacs, Fr., Grigorescu, S., Rădulescu, C., 1994]. Considerând transformarea realizată prin intermediul unei funcţii "T", vom putea scrie ecuaţia vectorială:

$$\vec{y} = T(\vec{x})$$

(1.3-2)

Funcţia unui sistem este ilustrată în Figura 1.3-1. In teoria sistemelor se demonstrează afirmaţia că orice sistem este o parte a unui sistem de rang superior, sau, altfel spus, că *orice parte a unui sistem este la rândul ei un sistem*. Admiţând deci această ierarhizare a sistemelor, se poate scrie relaţia lor de subordonare sub forma:

$$S^{(R-1)} \in S^{(R)}; S^{(R)} \in S^{(R+1)}$$

(1.3-3)

Se spune că între părţile de rang "R-1" ale unui sistem de rang "R" există o *relaţie sau o conexiune*, dacă cel puţin una din ieşirile unui subsistem este intrare pentru un alt subsistem

de acelaşi rang. Fie subsistemul 1, cu funcţia de transfer "T_1"şi subsistemul 2, cu funcţia de transfer "T_2". Pentru aceste două subsisteme se poate scrie:

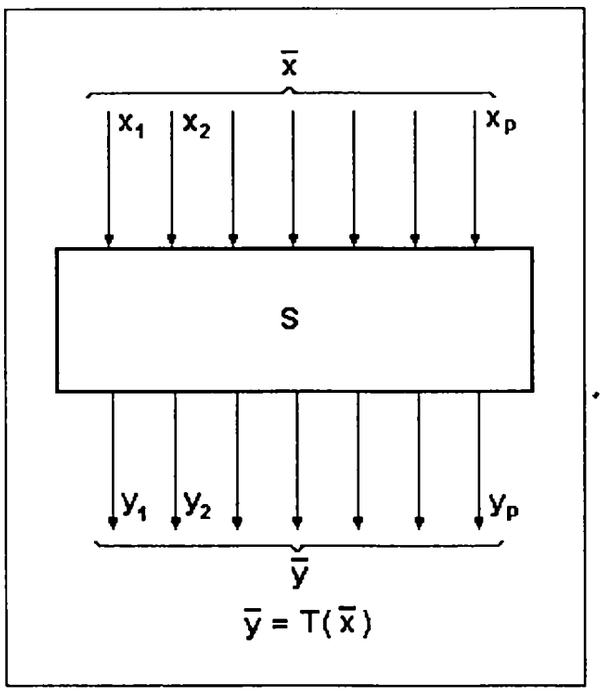


Figura 1.3-1

$$\vec{y}_{(1)} = T_{(1)}(\vec{x}_{(1)})$$
$$\vec{y}_{(2)} = T_{(2)}(\vec{x}_{(2)})$$
$$\vec{y}_{(1)} = (y_{(1)1}, y_{(1)2}, \ldots, y_{(1)q})$$
$$\vec{x}_{(2)} = (x_{(2)1}, x_{(2)2}, \ldots, \vec{x}_{(2)p})$$

(1.3-4)

Dacă între cele două subsisteme există o conexiune, atunci cel puţin pentru un "*p*" şi un "*q*" dat este adevărată relaţia:

$$x_{(2)p} = y_{(2)q}$$

(1.3-5)

Dacă se construieşte o matrice dreptunghiulară de dimensiuni (p×q) în care elementele au valoarea 1 sau 0 după cum relaţia (1.3-5) este sau nu este adevărată, se obţine matricea de cuplare a celor două subsisteme K_{12}:

$$K_{12} = \begin{bmatrix} e_{11} & e_{12} & e_{1p} \\ e_{21} & e_{22} & e_{2p} \\ \ldots & \ldots & \ldots \\ e_{q1} & e_{q2} & e_{qp} \end{bmatrix}$$

(1.3-6)

Procedura de obţinere a matricei de cuplare este exemplificată în Figura 1.3-2. Pe baza legăturilor dintre cele două subsisteme se obţine matricea de cuplare de forma:

$$K_{12} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(1.3-7)

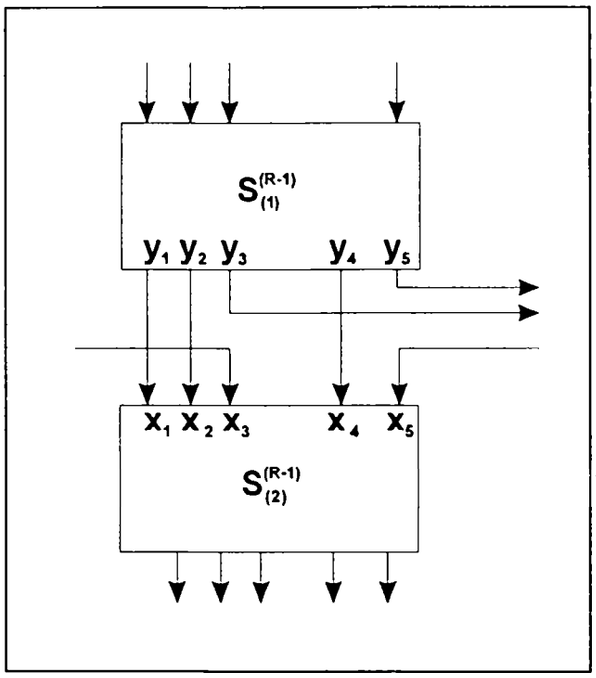


Figura 1.3-2

Legătura dintre cele două subsisteme de rang "R-1" mai poate fi exprimată şi prin următoarea ecuaţie vectorială:

$$\vec{x}_{(2)} = \vec{y}_{(1)} \cdot K_{12}$$

(1.3-8)

Prin generalizarea acestui procedeu la toate cele "N" subsisteme de rang "R-1" ale unui sistem de rang "R", se pot obţine cel mult N(N-1) matrici de cuplare, considerând că un subsistem nu poate fi cuplat cu el însuşi. Cuplările dintre subsisteme se pot realiza în ambele sensuri, existând deci atât o matrice de cuplare "K_{ij}"cât şi o matrice "K_{ji}", în general diferite. Evident că atunci când subsistemul "*i*" nu este cuplat cu subsistemul "*j*" matricea "K_{ij}" va fi nulă. Se poate construi astfel o matrice caracteristică întregului sistem, numită *matricea de structură "S"*, care evidenţiază în mod global toate legăturile care există în interiorul sistemului dat între subsistemele componente. In Figura 1.3-3 este redat un exemplu de sistem cu 6 subsisteme, pentru care a fost scrisă matricea de structură (1.3-9):

$$S = \begin{bmatrix} 0 & K_{12} & K_{13} & K_{14} & 0 & 0 \\ K_{21} & 0 & K_{23} & 0 & K_{25} & K_{26} \\ 0 & 0 & 0 & 0 & 0 & K_{36} \\ 0 & 0 & 0 & 0 & 0 & K_{46} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{64} & K_{65} & 0 \end{bmatrix}$$

(1.3-9)

In mod formal matricea de structură probează existenţa sau inexistenţa unor relaţii funcţionale între subsistemele aceluiaşi sistem, fără să se refere însă în nici un fel la aspectele calitative sau cantitative ale acestor relaţii. Acestea din urmă sunt legate mai ales de natura părţilor sistemului şi de modalitatea de organizare în spaţiu şi timp a acestuia. Se pot distinge astfel două aspecte de organizare ale unui sistem:

- *organizarea constructivă* a sistemului, legată de dispunerea în spațiu a părților și de aspectele particulare ale legăturilor determinate de topologia acestuia;
- *organizarea funcțională*, constituită mai ales din relațiile temporale care se constituie între părțile sistemului.

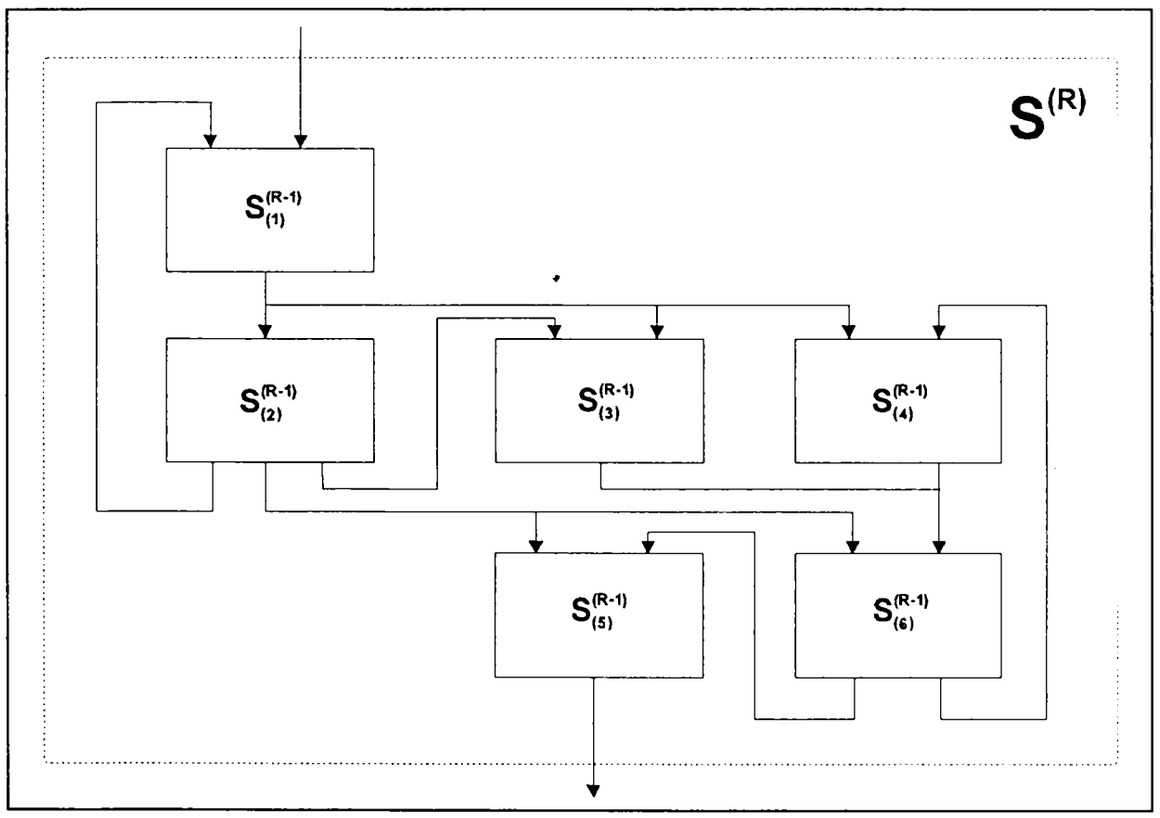


Figura 1.3-3

1.4. Sisteme de fabricație

In literatura de specialitate se duce încă o polemică susținută în jurul semnificației termenilor de "sistem de producție" și "sistem de fabricație". Conform definiției din [Ropohl, G., 1971, pag. 125], un *sistem de fabricație* este "*O mulțime de elemente tehnice și în același timp o mulțime de relații care se stabilesc între aceste elemente, care împreună determină obținerea unui produs finit, respectiv a unei familii de produse finite înrudite între ele*". Spre deosebire de acesta, un *sistem de producție* poate fi considerat "*orice sistem la a cărui ieșire se obține un produs, fie el finit sau semifabricat, iar în interiorul sistemului se găsesc diverse subsisteme de prelucrare sau de asamblare*". Definită în acest mod, noțiunea de "sistem de producție" este mai cuprinzătoare decât cea de "sistem de fabricație".

Pentru a putea descrie din punct de vedere calitativ funcția unui sistem, trebuiesc determinate intrările și ieșirile cele mai importante ale acestuia. Din acest punct de vedere, "*un sistem este un mijloc, un procedeu sau o schemă care se desfășoară în timp în conformitate*

cu un anumit scop" [Ropohl, G., 1971]. Funcţia sau rolul acestui sistem, este de a modifica într-un proces cronologic intrările sale sub formă de informaţie, energie sau materie şi de a furniza la ieşire acelaşi tip de elemente. Ţinând cont că atât informaţia vehiculată într-un sistem cât şi energia şi materia au parametrii cantitativi care se modifică în timp, se poate vorbi de fluxuri de materie, energie sau informaţie. *Deci, funcţia unui sistem este de a controla fluxurile de energie, informaţie şi materie care se stabilesc la intrarea şi la ieşirea acestuia.*

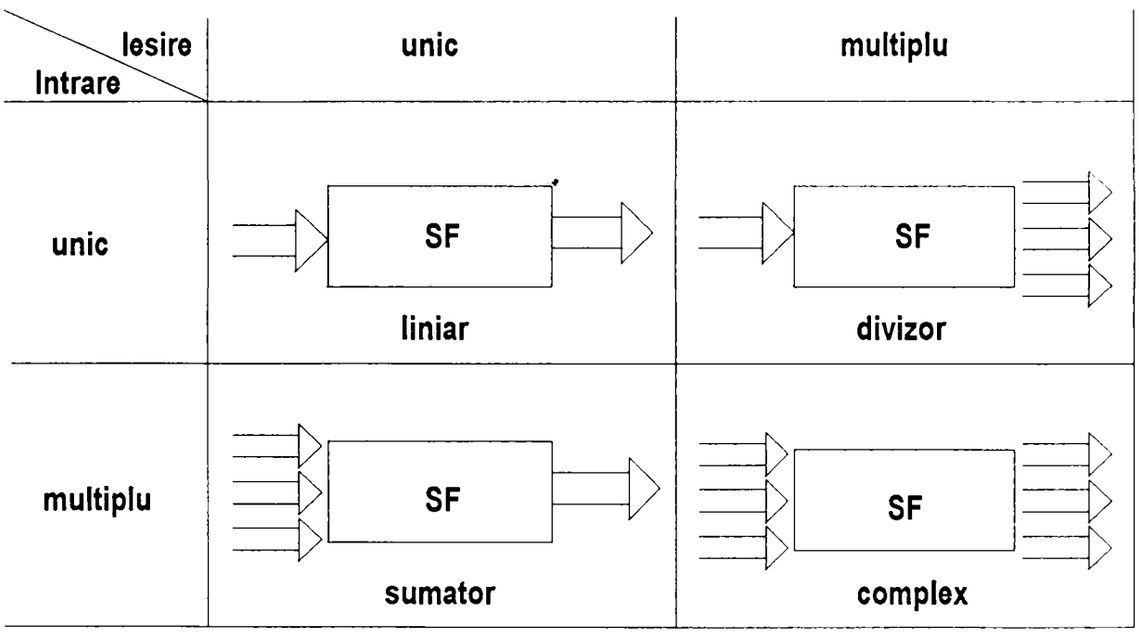


Figura 1.4-1

Fluxul de materiale constituie atât un element de intrare cât şi un element de ieşire al unui sistem de fabricaţie. Aceste fluxuri pot fi singulare sau multiple, rezultând de aici diversele tipuri de combinaţii, după cum se poate observa în Figura 1.4-1. Fiecare flux de intrare şi ieşire este caracterizat prin tipul de material, starea acestuia şi de rata sau indicele de consum, care reprezintă variaţia în timp a cantităţii de material de un anumit tip.

Din punct de vedere al fluxurilor de materiale, sunt valabile următoarele două principii [Ropohl, G., 1971]:

- principiul *continuităţii calitative*, care afirmă că proprietăţile fizice şi chimice ale materialelor vehiculate în sistem rămân neschimbate;
- principiul *continuităţii cantitative*, care afirmă că volumul total al materialelor intrate în sistem trebuie să egaleze volumul total de materiale ieşite din sistem într-un anumit interval de timp, care nu întotdeauna admite o minimalizare infinită.

Conform acestor afirmaţii sistemul ar trebui să conserve materia sub toate aspectele ei, ceea ce poate fi adevărat doar la sistemele simple de prelucrare prin aşchiere sau la sistemele de

montaj. Dacă ne referim însă la procese cu transformări chimice sau fizico-chimice, atunci cele două principii trebuiesc formulate în concordanță cu principiile generale de conservare a materiei.

Se presupune că dacă se face referire la sisteme de fabricație prin prelucrări mecanice sau la sisteme de montaj şi se notează cu "ΔV_i" volumul total de material de la intrare şi cu "ΔV_e" cel de la ieşirea din sistem, ambele raportate la un interval de timp "Δt", atunci principiul continuității cantitative poate fi exprimat analitic astfel:

$$\frac{\Delta V_e}{\Delta t} = \frac{\Delta V_i}{\Delta t}$$

(1.4-1)

Printre elementele de intrare sau de ieşire mai deosebite, în cazul sistemelor de fabricație, trebuiesc luate în considerare şi *materialele auxiliare*, cum ar fi lichidele de răcire pentru scule şi semifabricat, sau lubrifianții utilizați în sistem. Ca materiale de ieşire trebuiesc considerate inclusiv reziduurile sub formă de şpan sau rebuturile prognozate ale sistemului. Sculele aşchietoare sunt considerate ca elemente de structură ale sistemului, din acest motiv schimbarea lor nu este considerată un aport material în sistem, ci o modificare a structurii acestuia.

Fluxul de energie este indispensabil oricărui sistem de producție în care au loc transformări materiale sau informaționale. De regulă în sistemele de fabricație forma primară de energie folosită este energia electrică. In cazul în care unele subsisteme folosesc alte forme de energie, aceasta este preparată local cu ajutorul unor instalații adecvate, ele însele fiind părți componente ale sistemului general şi funcționând tot pe bază de energie electrică.

Fluxul de informații care se constituie în interiorul unui sistem de fabricație se supune teoriei matematice generale dezvoltate de Hartley şi ulterior de Shannon, Weaver şi Wiener. Informația circulă întotdeauna între un emițător şi un receptor, pe baza unei scheme reprezentată în Figura 1.4-2. Informația are sens şi poate fi apreciată ca atare din punct de vedere semantic numai dacă intersecția dintre semnalul emis şi cel receptat diferă de mulțimea vidă.

Pentru a putea aprecia aspectele cantitative şi calitative ale circulației informației într-un sistem flexibil, va trebui să pornim de la definiția noțiunii de *informație* şi să luăm de asemenea în considerare câteva aspecte intrinseci ale gradului de nedeterminare al unui eveniment ca rezultat al unui experiment. [Skach, C., 1985, pag. 3 şi urm.].

Informația se consideră a fi semnificația atribuită unui şir de date prin intermediul convențiilor utilizate pentru a le reprezenta. Datele sunt transferate cu ajutorul unor *semnale*, care sunt mărimi fizice cu o evoluție dependentă de timp.

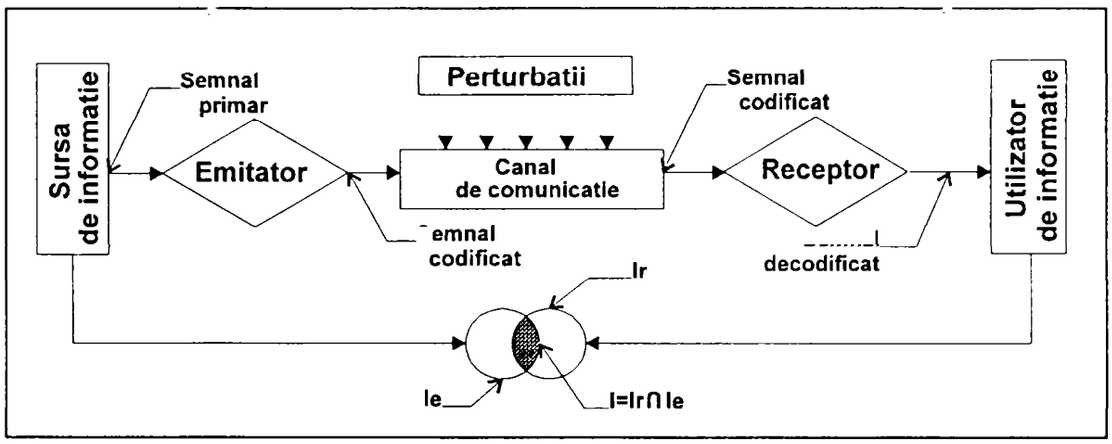


Figura 1.4-2

Informația este strâns legată de desfăşurarea evenimentelor, prin următoarele aspecte semantice:

- informația trebuie să aibă întotdeauna *un anumit sens*, să înlăture o anumită nedeterminare;
- în cazul unor informații diferite, cu conținut comparabil de sensuri, informația care *înlătură numărul cel mai mare de nedeterminări* are importanța cea mai mare;
- informația poate fi interpretată numai dacă este *materializată* printr-un anumit fenomen sau proces fizic, lucru ce se realizează prin semnale;
- informația are sens numai dacă există o *relație de comunicație* între sursă şi receptor.

Pentru transmiterea informației, o sursă utilizează în general un anumit *limbaj*, format dintr-o *reuniune de semne şi reguli semantice*. In sensul dat de Shannon [Shannon, G., E., 1948], *cantitatea de informație* a unui şir de "k" caractere din cele "n" caractere folosite pentru comunicare, fiecare având probabilitatea de apariție "p_i" este:

$$I = -C\sum_{i=1}^{k} p_i \log_a p_i$$

(1.4-2)

Unitatea de măsură utilizată pentru cantitatea de informație este legată de baza "a" a logaritmului. Dacă se consideră a=2 şi C=1 se obține "Bit"-ul, deci se va putea scrie:

$$I = -\sum_{i=1}^{k} p_i \log_2 p_i \quad [\text{Bit}]$$

(1.4-3)

Cantitatea maximă de informație "I_{max}" se va obține atunci când sunt folosite toate cele n caractere sau semne şi acestea au probabilități egale de apariție, deci sunt *echiprobabile*. In acest caz se va putea scrie:

$$I_{max} = \log_2 n \ [Bit]$$

(1.4-4)

De asemenea se poate defini *cantitatea de decizie* "H_o" de care este nevoie pentru a selecta un caracter dintr-un set de n caractere distincte şi echiprobabile, egală cu cantitatea de informație pe care o oferă un element selectat:

$$H_o = \log_2 n \ [Bit]$$

(1.4-5)

Pentru a aprecia calitativ informația oferită de un anumit eveniment, în sensul apropierii ei de cantitatea maximă de informație posibilă într-un caz dat, se determină *redundanța* "R" a evenimentului respectiv cu relația:

$$R = \frac{I_{max} - I}{I_{max}}$$

(1.4-6)

Spre exemplificare se presupune că sursa de informație foloseşte pentru comunicare "cuvinte" de lungime fixă, formate de exemplu dintr-un număr "k" de caractere echiprobabile. In aceste condiții se vor putea obține cu cele "n" elemente ale setului de caractere (alfabet) grupate câte "k", un număr de "n^k" combinații distincte, ceea ce înseamnă că pentru selectarea unui element al acestei mulțimi de cuvinte va fi necesară cantitatea de decizie:

$$I_{Cmax} = \log_2 n^k = k \log_2 n = k I_{max}$$

(1.4-7)

Se va putea defini şi în acest caz o redundanță a selecției unui cuvânt, care va avea valoarea:

$$R_C = \frac{I_{Cmax} - I_C}{I_{Cmax}}$$

(1.4-8)

Pentru a evidenția importanța practică pe care o are această teorie, se presupune că într-o magazie sunt 8 locaşuri de depozitare, (vezi Figura 1.4-3), care pot fi pline sau goale. Numărul combinațiilor posibile va fi deci 2^8 şi cantitatea de decizie necesară pentru a selecta un anumit element va fi:

$$H_0 = \log_2 2^8 = 8 \text{ [Bit]}$$

(1.4-9)

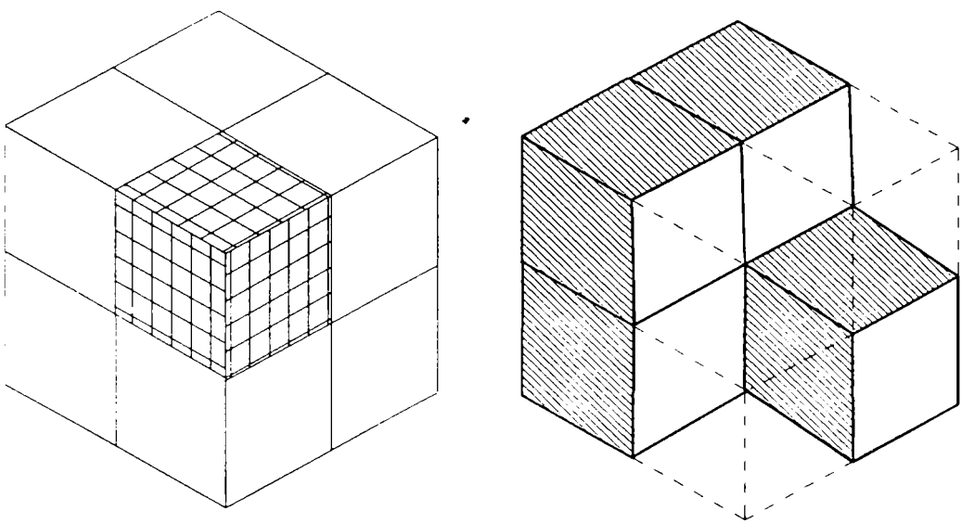


Figura 1.4-3

Dacă toate locaşele sunt pline sau goale, atunci alegerea unui element presupune o cantitate nulă de informație sau de decizie, pentru că numărul cazurilor posibile este 1, orice alegere fiind considerată un eveniment. Dacă în schimb un număr de 3 locaşuri sunt fără material, deci sunt goale, atunci cantitatea de informație pe care o conține selecția unui anumit element va fi (conform ecuației (1.4-7)):

$$H = -8(5/8\log_2 5/8 + 3/8\log_2 3/8) = 7.64 \text{ [Bit]}$$

(1.4-10)

In relația de mai sus fracțiile 3/8 şi 5/8 reprezintă probabilitățile de alegere a unui locaş gol, respectiv plin.

Dacă se presupune acum că se prelucrează o piesă cubică aşa cum este cea din Figura 1.4-3, atunci se poate observa că la intrarea în sistemul de fabricație piesa neprelucrată avea un conținut informațional nul iar după prelucrare, care a constat în îndepărtarea celor trei cuburi, piesa are un conținut informațional de 7.64 [Bit].

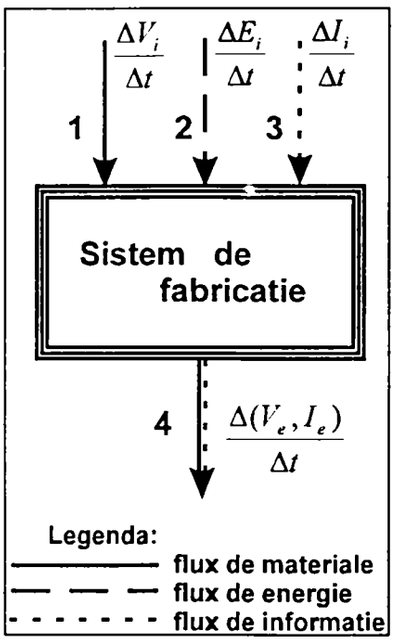


Figura 1.4-4

In concluzie, la ieşirea unui sistem de fabricaţie nu se obţine doar un obiect material oarecare, ci un obiect purtător de informaţie. Cele două fluxuri, material şi informaţional, sunt suprapuse, constituind astfel un singur flux. Pornind de la aceste observaţii, cu notaţiile din Figura 1.4-4, funcţia unui sistem de fabricaţie se poate scrie astfel:

$$\frac{\Delta(V_e, I_e)}{\Delta t} = T\left(\frac{\Delta V_i}{\Delta t}, \frac{\Delta E_i}{\Delta t}, \frac{\Delta I_i}{\Delta t}\right) \qquad \textbf{(1.4-11)}$$

In situaţiile cele mai simple, un sistem poate îndeplini o singură funcţie, deci transformă în mod unic fluxurile de intrare într-un singur flux de ieşire, cu caracter mixt, material şi informaţional, aşa cum s-a arătat mai sus. Acestea sunt *sistemele rigide, neflexibile*. Dacă însă sistemul poate îndeplini mai multe funcţiuni, astfel încât să se poată obţine din mai multe fluxuri de intrare mai multe fluxuri de ieşire, atunci se spune că sistemul este *flexibil*. Aceasta este situaţia cea mai avantajoasă pentru un sistem de fabricaţie, deoarece oferă cele mai largi posibilităţi de utilizare a capacităţii de producţie a maşinilor unelte şi a celorlalte dispozitive şi maşini de lucru din cadrul sistemului. Evident că această afirmaţie rămâne să fie dovedită la nivel teoretic. La nivel practic ea a fost pe deplin verificată de performanţele pe care le obţin sistemele flexibile de fabricaţie în peisajul industrial contemporan, unde ideea de *adaptabilitate maximă la cerinţele pieţei* este unul din criteriile de performanţă de prim ordin.

Conform celor prezentate în subcapitolele precedente, se poate afirma că un *sistem de fabricaţie* este mulţimea unor subsisteme orientate spre producţia unor elemente materiale, împreună cu relaţiile care se stabilesc între ele pentru realizarea scopului propus. Componentele (subsistemele) esenţiale ale unui sistem de fabricaţie precum şi relaţiile care se stabilesc între ele, în conformitate cu Figura 1.4-5, sunt:

- *subsistemul de lucru 1*, care are la intrare, în mod necesar, fluxuri de materiale, energie şi informaţie, iar la ieşire un flux combinat, material şi informaţional;
- *subsistemul de acţionare 2*, care are la intrare fluxul de energie şi fluxul de informaţie, iar la ieşire fluxuri de energie eventual de diverse tipuri, pentru ceilalţi consumatori din sistem;

- *subsistemul de manevră 3*, la intrarea căruia se găsesc fluxuri de energie şi de informaţie, iar la ieşire fluxuri de informaţie pentru subsistemul de lucru şi fluxuri de informaţie de control a poziţiei momentane a componentelor pentru subsistemul general de conducere;
- *subsistemul de conducere 4*, prevăzut la intrare cu fluxuri energetice şi informaţionale de la toate subsistemele componente ale sistemului dat, iar la ieşire cu fluxuri informatice pentru subsistemele de acţionare, mişcare şi manipulare;
- *subsistemul de manipulare 5*, care are la intrare fluxuri de toate genurile, materiale brute sau prelucrate, energie şi informaţie, iar la ieşire doar fluxuri materiale, cu şi fără conţinut informaţional;
- *subsistemul de măsurare şi control 6,* la intrarea căruia se găseşte un flux de material prelucrat care provine de la subsistemul de lucru iar la ieşire tot un flux de material prelucrat şi un flux informaţional care este destinat sistemului de conducere.

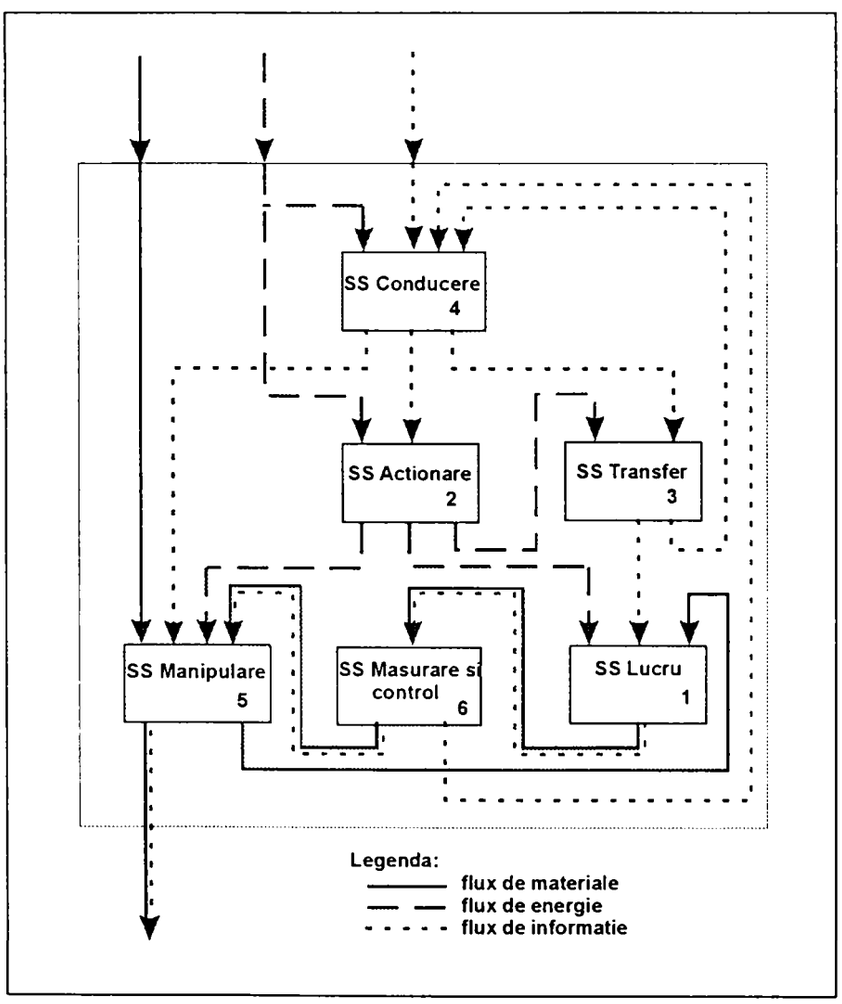


Figura 1.4-5

Matricea de structură a sistemului, care conţine toate matricile de legătură (sau de cuplare) dintre părţi, poate fi scrisă în felul următor:

$$S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & K_{16} \\ K_{12} & 0 & K_{23} & 0 & K_{25} & 0 \\ K_{31} & 0 & 0 & K_{34} & 0 & 0 \\ 0 & K_{42} & K_{43} & 0 & K_{45} & 0 \\ K_{51} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{64} & K_{65} & 0 \end{bmatrix}$$

(1.4-12)

Structura sistemului prezentat în Figura 1.4-5 trebuie considerată o structură minimală, orientată spre realizarea unei anumite sarcini de producție şi anume obținerea unui produs finit pornind de la materia primă. Acest lucru se poate realiza uneori pe o singură maşină, dar de obicei este rezultatul cumulat al mai multor mijloace de producție, fiecare executând o anumită fracțiune din sarcina globală de producție. *Sarcinile de producție parțiale au acelaşi rang logic ca şi cea globală*, deoarece nerealizarea unei sarcini parțiale atrage după sine compromiterea întregului program. Fiecare din mijloacele de producție participante modifică fluxul de materiale din sistem sub aspectul formei, folosind un flux de energie. Pentru fiecare sarcină de producție poate fi considerat un sistem cu o structură identică cu a sistemului global. Un asemenea sistem, orientat spre realizarea unei sarcini de producție parțiale şi care are o structură identică cu sistemul global, poartă denumirea de *"cosistem"*.

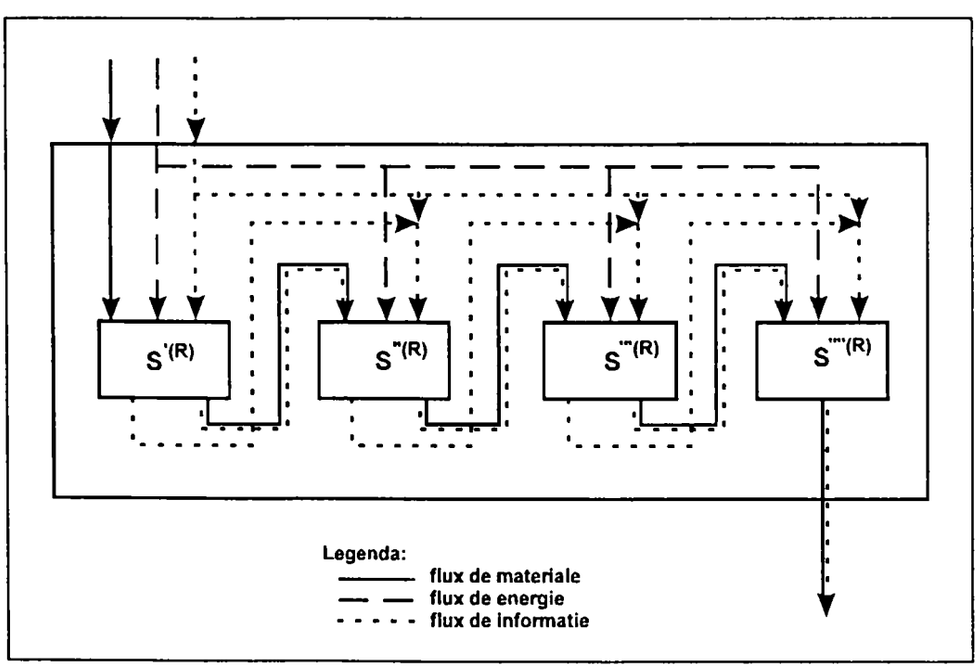


Figura 1.4-6

In Figura 1.4-6 este reprezentat un sistem format dintr-o *succesiune de cosisteme*, care au structură identică şi sunt alimentate central cu fluxuri de energie şi informaţie. In ce priveşte fluxul de materiale, acesta se constituie prin înseriere, în sensul că fluxul de ieşire de natură material-informaţională al unui cosistem, este considerat flux de intrare de aceeaşi natură pentru cosistemul următor. Se observă astfel că informaţia se transferă şi se modifică de la un cosistem la altul după o anumită logică, fapt care determină *necesitatea ca toate subsistemele de conducere ale cosistemelor să fie conectate la un subsistem de conducere central*.

1.5. Analiza fluxului de informaţie

Din punct de vedere al fluxului de informaţie, se disting două aspecte importante care definesc caracterul acestuia:

- toate informaţiile legate de modificările dimensionale şi de formă ale materialului se constituie în *fluxul informaţiilor de formă*;
- toate informaţiile legate de tehnicile şi procedeele de modificare a formei sau stării materialului constituie *fluxul informaţiilor tehnologice*.

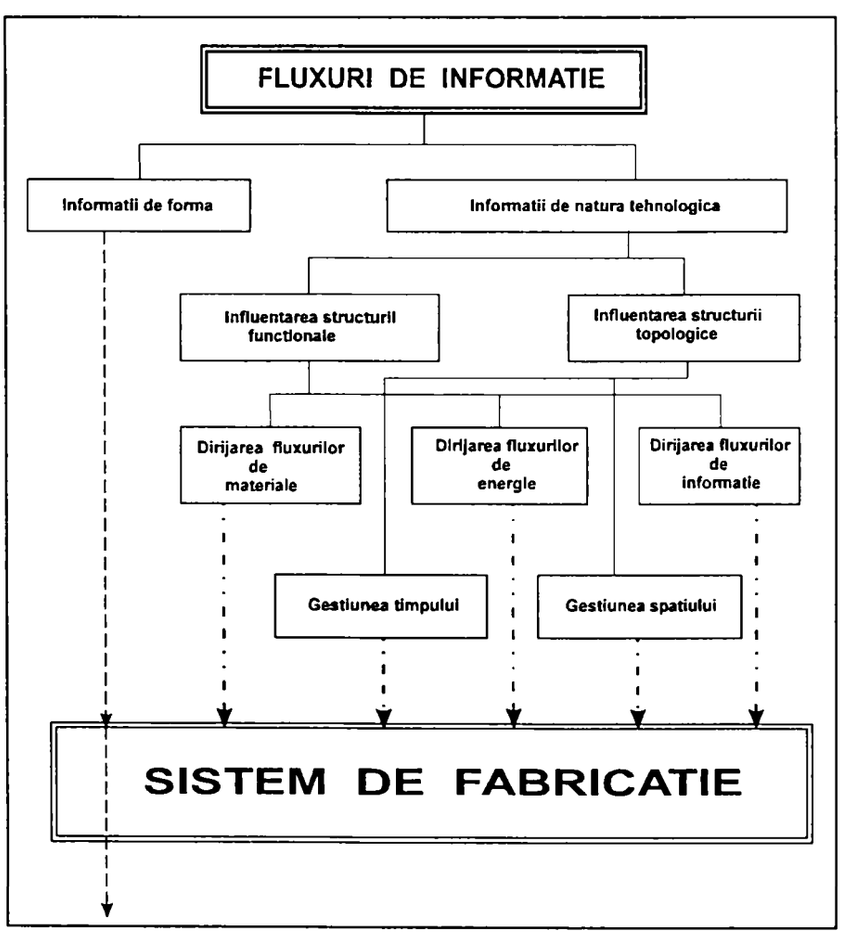


Figura 1.5-1

Aşa după cum se poate observa în Figura 1.5-1, *informaţiile de formă* se constituie într-un flux a cărui localizare este la nivelul sistemului de fabricaţie, unde se produce de altfel modificarea parametrilor acestuia.

Fluxul informaţiilor tehnologice are două componente importante, care influenţează pe de o parte structura funcţională a sistemului şi pe de altă parte structura sa topologică:

- din punct de vedere al *structurii funcţionale*, fluxul informaţional este implicat în dirijarea fluxului de materiale, a fluxului de energie şi în propria sa gestiune şi dirijare, adică a fluxului de informaţie;
- cealaltă componentă, referitoare la *influenţa asupra topologiei sistemului*, cuprinde gestiunea desfăşurării în timp a proceselor şi conducerea ordonării spaţiale.

Prima categorie de informaţii, cele de formă, se află stocate în materialul care circulă în sistem. Celelalte categorii care au fost menţionate mai sus se află stocate codificat sau în clar pe diverşi purtători de informaţie, cu acces aleator sau serial, în formă "read-write" sau "read-only".

1.6. Programarea unui sistem de fabricaţie

In cadrul sintezei sistemelor de fabricaţie se pot distinge două categorii de acţiuni care au ca rezultat de ansamblu sistemul fizic, capabil să îndeplinească un anumit grup de sarcini de producţie:

- *elaborarea structurii sistemului*, ca şi componentă hard a sintezei;
- *elaborarea programului de conducere* ca şi componentă soft.

In funcţie de raportul care se stabileşte între cele două laturi ale acţiunii de sinteză a sistemelor de fabricaţie flexibilă, se pot identifica mai multe procedee de programare, care vor fi prezentate în continuare.

In esenţă, *programarea unui sistem de fabricaţie flexibilă* poate fi definită ca fiind aspectul informaţional al activităţii de sinteză, adică:

- formularea detaliată şi precisă a unei probleme de rezolvat;
- stabilirea unui algoritm corespunzător ca regulă de rezolvare;
- stocarea algoritmului şi a descrierii problemei;
- transmiterea informaţiilor către sistem.

Toate aceste operaţii de programare descrise mai sus se pot regăsi într-o memorie externă sau într-un dispozitiv intern de stocare a informaţiilor, care face parte din însăşi

structura sistemului. Activitatea de programare presupune un flux informațional care este reprezentat în Figura 1.6-1. Sistemul de programare utilizează mai multe categorii de informații, stocate pe suport intern sau extern sistemului. Aceste informații provin din bazele de date ale sistemelor CAD/CAM şi se referă la forma şi tehnologia de execuție a reperelor ce se prelucrează în sistem, sau provin de la subsistemele sistemului, caz în care au semnificația unor legături de reacție care să permită urmărirea funcționării în timp real a sistemului flexibil de fabricație.

Sarcinile de programare (de conducere) într-un sistem de fabricație se pot împărți în două mari categorii: sarcini de conducere la nivel local (la nivelul controlerelor locale din fiecare celulă sau nod al rețelei) şi sarcini de conducere la nivel global (la nivelul calculatorului central care dirijează întreaga activitate din sistem).

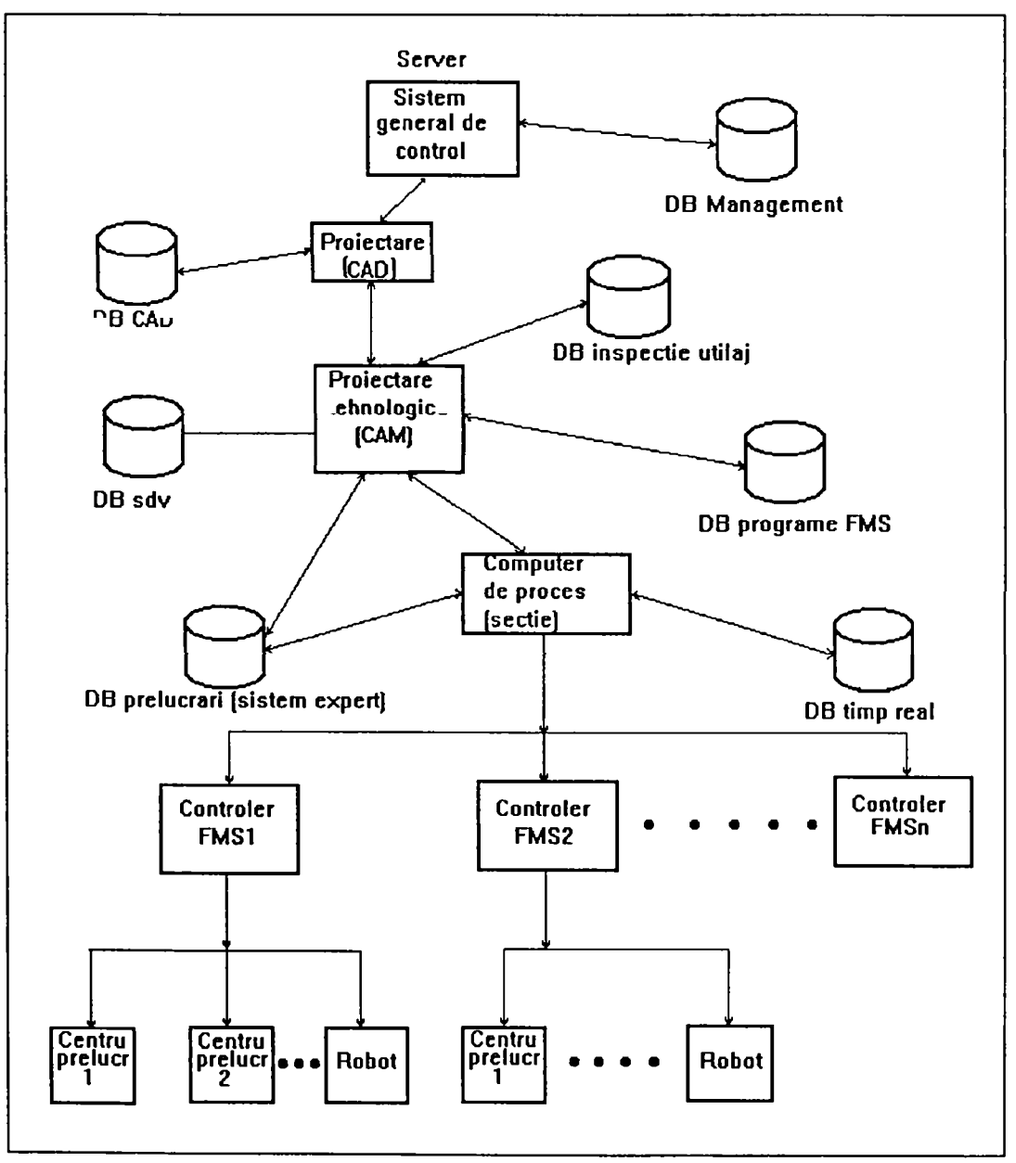


Figura 1.6-1

Programarea unui sistem de fabricație are drept finalitate elaborarea unui program de conducere care rulează într-un calculator de proces situat de obicei la nivelul secției de producție [Ranky, Dr. P., 1990, pag. 37 şi urm.]. In Figura 1.6-1 se poate observa organizarea generală a unui sistem de prelucrare a datelor într-un sistem flexibil de fabricație, precum şi locul pe care îl ocupă în acest ansamblu sistemul de conducere al celulei flexibile. In ierarhia presupusă de conducerea integrată a unei întreprinderi, sistemul de conducere al celulei flexibile trebuie să fie interconectat cu toate celelalte sisteme soft, inclusiv cu bazele de date statice sau dinamice cu care este prevăzută unitatea respectivă de producție.

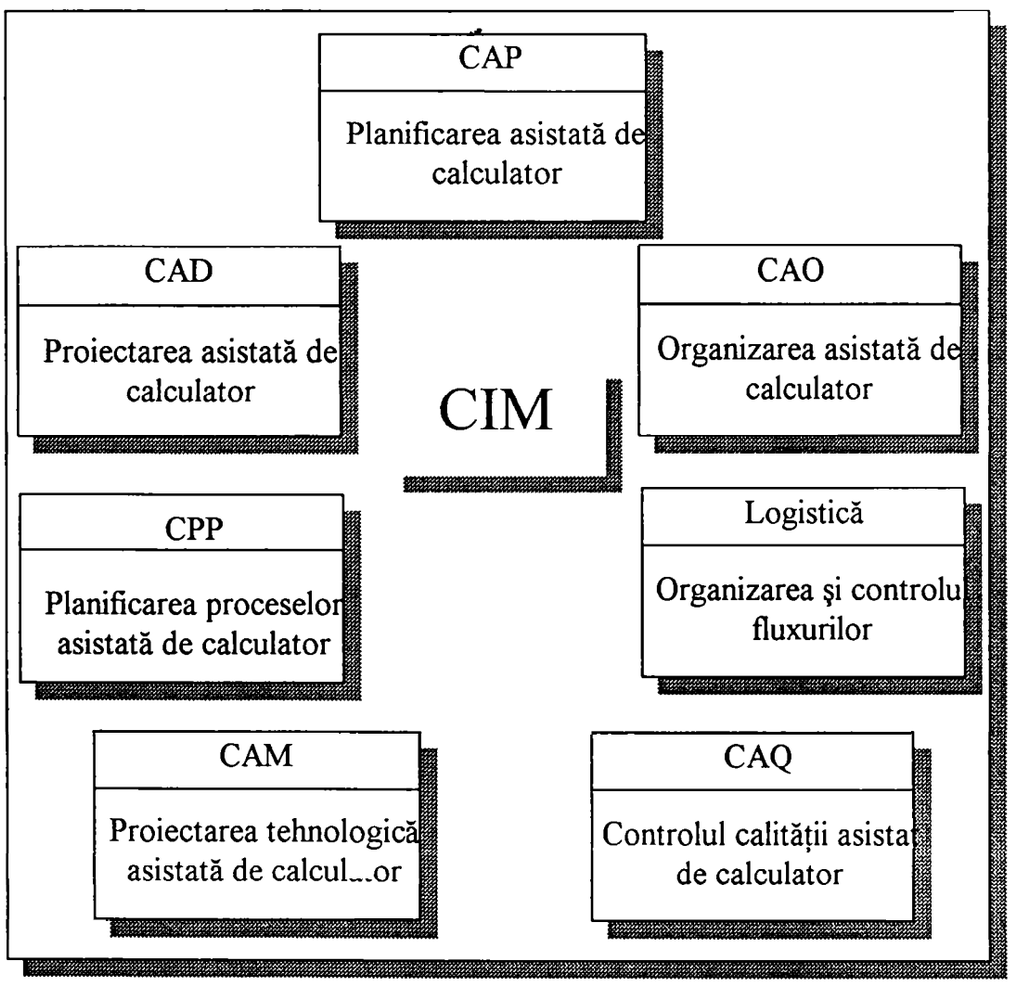


Figura 1.6-2

Conducerea prin calculator a procesului de fabricație într-un sistem flexibil trebuie să rezolve corect următorul inventar minimal de probleme:

- gestiunea timpului;
- selectarea programelor de fabricație pe repere;
- conducerea sistemului logistic;

- sesizarea funcționării incorecte a sistemului;

Această modalitate de conducere a sistemelor flexibile a devenit posibilă datorită nivelului atins în dezvoltarea actuală a sistemelor de calcul, precum şi datorită dezvoltării unor programe speciale care se ocupă de conducerea asistată de calculator a proceselor (CPP).

Pe linia utilizării computerelor în sistemele flexibile de fabricație, în speță a sistemelor CIM, în [VDI - Gemeinschaftsausschuß CIM, 1991, bild 2-2] este prezentată o schemă a componentelor unui astfel de sistem (vezi Figura 1.6-2).

Întreaga logistică utilizată în sisteme de fabricație este orientată spre reducerea timpilor tehnologici, a timpilor de pregătire şi spre reducerea stocurilor.

1.7. Elemente de analiză a fluxurilor de materiale

Subsistemul de manipulare, aşa cum a fost definit el în §1.4, este responsabil de mişcarea, stocarea, paletizarea sau controlul elementelor materiale din sistem cu rol de materie primă, semifabricate şi subansamble, elemente de prindere şi fixare, dispozitive diverse, scule şi alte elemente similare. In principal, problemele cele mai complexe din punct de vedere al transferului unor obiecte din sistem apar în legătură cu mişcarea pieselor de lucru. Un sistem de conducere şi control prin calculator a subsistemului de manipulare urmăreşte ca materialul de un anumit tip să fie adus la momentul potrivit şi în locul potrivit, în conformitate cu cerințele impuse de planul de producție al celulei sau al sistemului de fabricație. Iată câteva dintre *principiile de organizare ale unui sistem de manipulare a materialelor*, aşa cum rezultă ele din [Viswanadham, N., Narahari, Y., 1992, § 2.5 şi urm.]:

- cel mai bun sistem de manipulare a materialelor este cel care nu manipulează nimic;
- reducerea distanțelor de transfer prin gruparea utilajelor în arii cât mai reduse sau de-a lungul unor rute de transfer cât mai simple;
- obținerea unui timp de răspuns cât mai scurt, prin utilizarea unor rețele de transfer formate din utilaje rapide, de dimensiuni reduse şi echipate cu dispozitive de control inteligente;
- ordonanțarea efectivă a sarcinilor de transfer prin urmărirea stării momentane a tuturor utilajelor şi pieselor din sistem.

Clasificarea sistemelor de transfer poate fi urmărită în Figura 1.7-1.

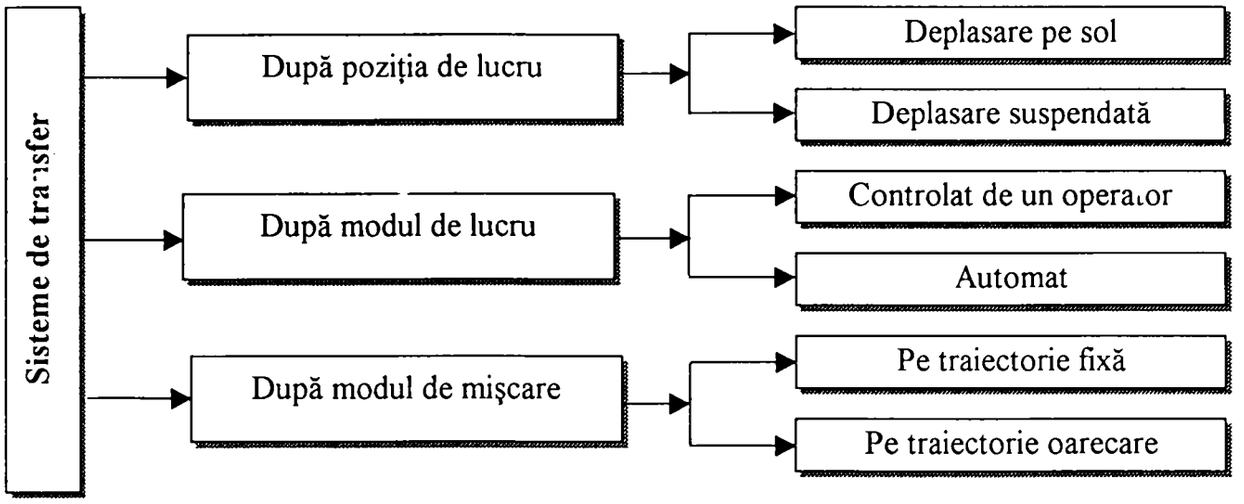


Figura 1.7-1

Sistemele de transfer industrial sunt de o foarte mare diversitate constructivă. Ele sunt constituite dintr-o rețea de conveioare, vehicule ghidate automat (AGV), roboți industriali şi alte echipamente similare. Unele dintre ele, cum ar fi liniile transfer sau liniile carusel au *funcționare sincronă*, în sensul că toate materialele din sistem care se mişcă între diverse noduri ale rețelei de fabricație se deplasează simultan. Altele, de exemplu sistemele AGV, au *funcționare asincronă*.

Una din cerințele de maximă importanță ridicate în fața sistemelor de transfer a materialelor este *flexibilitatea*. Această cerință provine din nevoia de a acoperi trasee de deplasare foarte diverse, de lungimi şi configurații diferite, iar tipurile de materiale care trebuiesc transferate să poată fi şi ele diferite.

Conveioarele sunt dispozitive de transfer de flexibilitate redusă, atât sub aspectul rutei cât şi al tipurilor de piese transferate. Din punct de vedere constructiv se pot întâlni *conveioare cu o singură direcție de deplasare*, de tipul celui din Figura 1.7-2, respectiv *conveioare cu recirculare în buclă*, aşa cum se poate vedea în Figura 1.7-3.

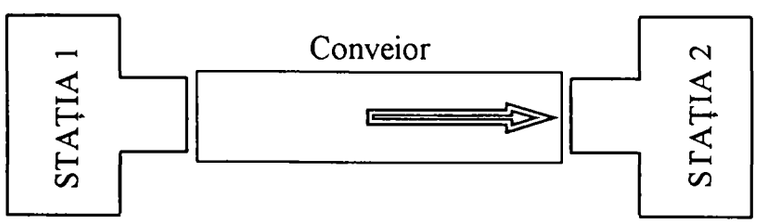


Figura 1.7-2

In legătură cu funcționarea unui conveior se foloseşte noțiunea de rată de alimentare "f", definită cu relația:

$$f = \frac{v}{d} [buc/min]$$

(1.7-1)

unde: v [m/min] - viteza de deplasare a conveiorului;

d [m] - distanța dintre două piese consecutive pe conveior.

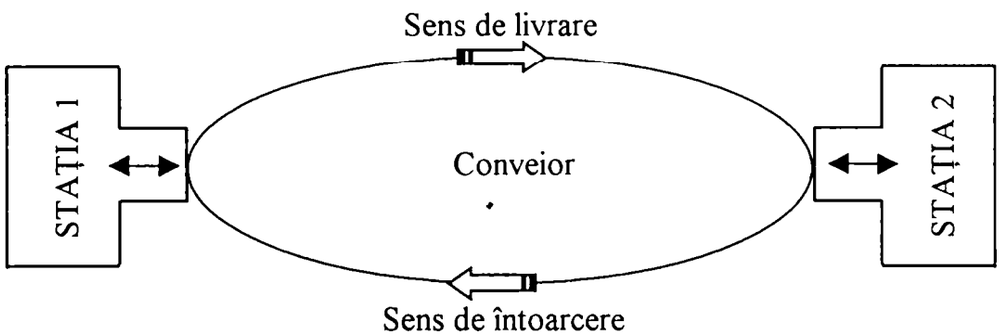


Figura 1.7-3

Roboții industriali folosiți ca dispozitive de transfer sau manipulare se caracterizează printr-o mare flexibilitate din punct de vedere al tipurilor de piese transferate/manipulate, dar sunt mai puțin flexibili în ce privește traiectoria. De obicei roboții sunt folosiți la servirea unui grup de mașini sau de echipamente, așa cum se poate observa din Figura 1.7-4.

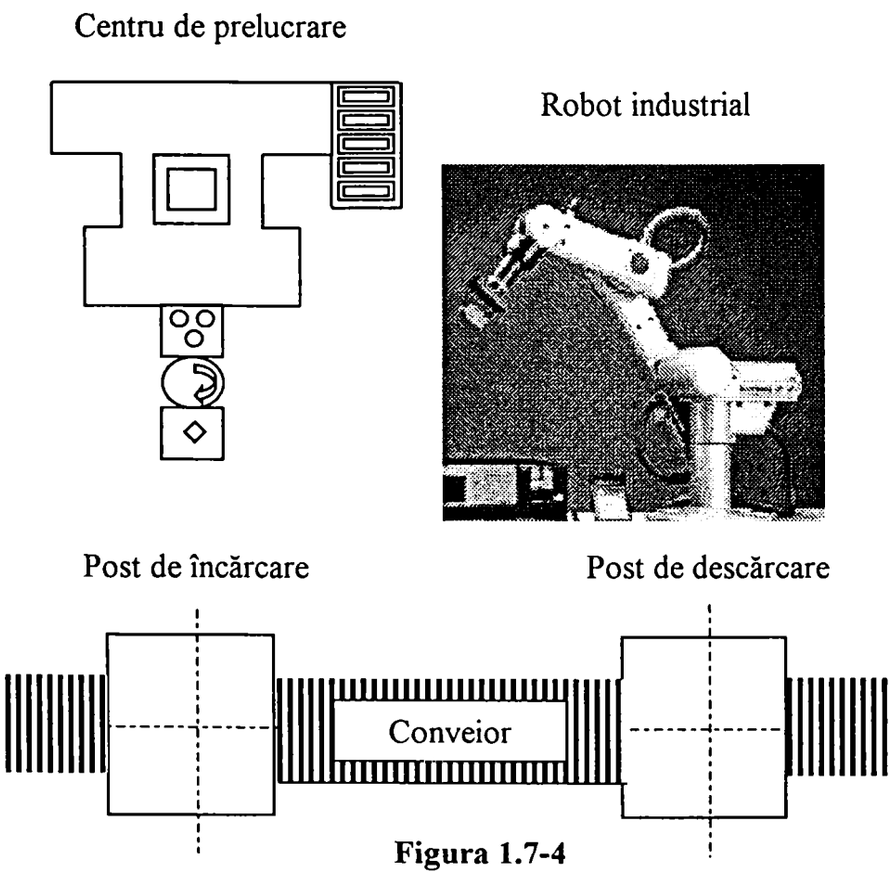


Figura 1.7-4

Utilizarea roboților pentru deplasarea materialelor presupune unele *măsuri de protecție*, cum ar fi verificarea identității piesei înaintea începerii procesului de prelucrare sau verificarea prezenței piesei la locul de încărcare/descărcare înaintea lansării comenzilor către robot. Mişcările de materiale din sistem sunt consumatoare de timp şi au influență asupra duratei ciclului de fabricație. Pentru exemplul din Figura 1.7-4 se presupune că timpul principal de aşchiere pentru centrul de prelucrare este de 24 secunde şi că timpul de întrerupere accidentală este de 10% din timpul total, adică 0,8 ore dintr-un schimb. In aceste condiții, dacă mişcările de materiale desfăşurate cu robotul sunt cele din Tabelul 1.7-1, durata unui ciclu de lucru va fi de 12s, iar productivitatea sistemului va fi:

$$P=7,2 \text{ ore} \times 3600 \text{ sec/oră} / (12+34) \text{ sec/piesă} = 720 \text{ piese/schimb}$$

Tabelul 1.7-1

Operația	**Durata totală 36s din care:**
Preluarea unei piese de pe conveior (inclusiv timpul de aşteptare pentru instalarea piesei în poziția de preluare)	2,6s
Deplasare cu robotul de la conveiorul de intrare până la maşină	1,7s
Încărcarea piesei pe maşină şi îndepărtarea brațului pentru a permite începerea lucrului	1,1s
Descărcare piesei de pe maşină	0,8s
Deplasare cu robotul de la maşină la conveiorul de ieşire	1,7s
Depozitarea piesei pe conveiorul de ieşire	0,3s
Deplasarea de la conveiorul de ieşire la cel de intrare	3,8s

Vehiculele ghidate automat (AGV) joacă un rol deosebit de important în organizarea fluxurilor de materiale dintr-un sistem de fabricație atunci când este vorba despre transferul materialelor pe orizontală. Spre deosebire de alte dispozitive şi sisteme de transfer, AGV-urile îşi pot alege singure traiectoria de deplasare pe baza unor criterii de optimizare sau îşi pot modifica dinamic această traiectorie pe baza unor informații legate de congestionarea traficului în diverse zone ale sistemului.

In funcție de sarcinile de transfer pe care le au în sistem, AGV-urile pot fi de mai multe tipuri:

- vehicule de remorcare;

- vehicule de transfer pentru piese unitare;
- vehicule de transfer pe linii de asamblare sincronă sau asincronă;
- vehicule purtătoare de braţe mecanice sau manipulatoare;
- vehicule ce deservesc sistemele flexibile;
- vehicule autonome, echipate cu controlere inteligente şi vedere artificială.

Indiferent de tipul lor, vehiculele de transfer în sisteme de fabricaţie pun problema comună a *modului de conducere şi de control a poziţiei* lor în sistem. Principalele tehnici de acest tip cunoscute până în prezent sunt următoarele:

- ghidarea *cu ajutorul câmpului electromagnetic* generat de un fir îngropat în pardoseală, are avantajul simplităţii şi dezavantajul rigidităţii traiectoriei. Este sistemul de ghidare folosit destul de frecvent;
- ghidarea pe baza *controlului mişcării de rotaţie a roţilor* de pe cele două părţi ale vehiculului, care poate da măsura razei de viraj a acestuia. Se foloseşte de regulă în conjuncţie cu ghidarea cu fir îngropat;
- *ghidarea optică* foloseşte de obicei marcaje din folii adezive vizibile pentru un sistem de senzori optici care operează în spectru vizibil sau UV. Este un sistem destul de flexibil, dar este sensibil la deteriorarea marcajelor;
- *ghidarea pe bază de imagini digitale* este un procedeu modern, bazat pe imaginea preluată cu camere video mobile sau fixe şi interpretată cu un sistem de prelucrare la nivel de pixeli care poate furniza în final semnale de comandă pentru controlerul vehiculului;
- *sistemele de ghidare inerţiale (sisteme giroscopice)* se bazează pe modificările apărute la nivelul parametrilor mişcării (acceleraţii), sesizate de un giroscop, atunci când apar modificări ale traiectoriei cu efecte inerţiale (accelerări sau frânări, mişcarea pe traiectorii circulare, etc.);
- ghidarea prin *metoda triangulaţiei* oferă posibilitatea mişcării în medii cu componenţă variabilă şi sunt insensibile la perturbaţiile mecanice sau luminoase uzuale. Metoda se bazează pe utilizarea unor surse de radiaţii, de obicei în infraroşu, amplasate pe periferie şi vizualizate de pe robocar. Este încă dificil de stăpânit multitudinea de date care trebuiesc prelucrate în timp real pentru o bună orientare şi ghidare a vehiculului;
- *folosirea ultrasunetelor şi a razelor laser* face de asemenea posibilă orientarea în medii variabile. Este nevoie de elemente de inteligenţă artificială pentru recunoaşterea obiectelor înconjurătoare, precum şi de capacităţi mari de prelucrare de date on-line.

Ordonanțarea mişcărilor unui vehicul de tip AGV se poate face în două moduri distincte:

- prin *alegerea unui vehicul din cele disponibile* pentru a prelua o sarcină de transfer;
- prin *atribuirea unor indici de prioritate* tuturor componentelor sistemului care pot emite cereri de transfer.

In continuare se vor prezenta câteva metode de ordonanțare a mişcărilor aplicabile în ambele situații prezentate anterior, mai întâi pentru cazul când *sistemul cuprinde mai multe vehicule şi cererea de transfer este unică*:

- metoda *selecției aleatoare* a unui vehicul din cele disponibile în sistem în momentul apariției cererii de transfer, indiferent de poziția acestuia fața de unitatea care formulează cererea;
- *regula vehiculului cel mai apropiat* determină alegerea vehiculului care se află cel mai aproape de unitatea care emite cererea de transfer, apropiere apreciată prin distanța minimă de parcurs sau prin timpul minim necesar deplasării vehiculului până la unitatea solicitantă;
- *regula vehiculului cu cea mai lungă aşteptare* determină alegerea acelui vehicul care staționează de cel mai mult timp;
- *regula vehiculului celui mai puțin încărcat* recomandă alegerea vehiculului care a efectuat cele mai puține misiuni de transfer până în momentul apariției cererii.

Aprecierea încărcării se poate face prin numărul de sarcini de transfer îndeplinite sau prin timpul de deplasare cu sarcină raportat la timpul total scurs în sistem.

In momentul în care *în sistem apar mai multe cereri de transfer simultan*, se pune problema de a găsi un algoritm pe baza căruia să fie alocată o cerere de transfer la un anumit vehicul din grup. Această operație se poate desfăşura în conformitate cu diverşi algoritmi:

- *selecția aleatoare* a unei sarcini de transfer, fără să se țină seama de nici un criteriu, este soluția banală a problemei, aplicabilă la sistemele cele mai simple;
- *alegerea cererii de transfer de la unitatea cea mai apropiată* are avantajul parcurgerii unor distanțe minime fără sarcină, dar pe de altă parte poate conduce la blocarea maşinilor mai îndepărtate, care nu vor fi deservite niciodată şi care au nevoie din acest motiv de depozite intermediare foarte încăpătoare;
- *alegerea unității celei mai îndepărtate* este opusă celei anterioare;

- algoritmul bazat pe *cea mai mare coadă de aşteptare* va alege pentru deservire dintr-o multitudine de cereri pe cea pentru care numărul de piese în aşteptare este maxim;
- algoritmul bazat pe *timpul petrecut de diverse repere în sistem* va alege pentru a fi deservită acea sarcină de transfer care provine de la piesa cu cel mai lung timp petrecut în sistem, favorizând astfel scurtarea timpilor de fabricaţie ai reperelor;
- în contradicţie cu algoritmul de mai sus se află algoritmul bazat pe *cel mai scurt timp petrecut în sistem*;
- algoritmul bazat pe *timpul de prelucrare cel mai scurt rămas până la terminarea prelucrării* favorizează de asemenea eliminarea din sistem a pieselor care sunt cel mai aproape de final.

Alegerea uneia din metodele de ordonanţare a sarcinilor de transfer se poate face analitic sau prin simulare, altfel fiind destul de dificil de apreciat efectul cumulat al diverselor variabile ale sistemului asupra indicilor de performanţă. *Unul din obiectivele prezentei lucrări este de a studia efectul acestor strategii de ordonanţare a mişcărilor de transfer asupra indicilor de performanţă tehnologică ai sistemului.*

1.8. Probleme speciale ale funcţionării sistemelor flexibile de fabricaţie

Anumite particularităţi de construcţie sau de funcţionare ale unui sistem de fabricaţie pot determina un comportament aparte al acestuia. *Filozofia de funcţionare a unui sistem constă în îndeplinirea sarcinilor de fabricaţie în mod continuu atât timp cât nu se modifică condiţiile minimale necesare funcţionării*. In esenţă aceste condiţii se referă la existenţa şi funcţionalitatea fluxurilor sistemului. Acestea impun ca pentru fiecare sarcină din ciclul de producţie să existe cel puţin o resursă disponibilă în sistem care să o poată prelua şi duce la îndeplinire. In funcţie de diversele situaţii care pot apare relativ la *integritatea fluxurilor*, se pot întâlni următoarele situaţii:

- *mai multe componente ale sistemului solicită simultan o anumită resursă din sistem*, caz în care apare un *conflict* care va trebui arbitrat într-un anumit fel;
- *mai multe resurse disponibile pot îndeplini aceeaşi sarcină de fabricaţie* la un moment dat în sistem, caz în care se vorbeşte despre *convergenţa resurselor*;
- din diverse motive, care duc la *întreruperea unuia din fluxurile vitale ale sistemului*, acesta încetează să-şi mai îndeplinească rolul funcţional, caz în care vorbim de un *blocaj*;

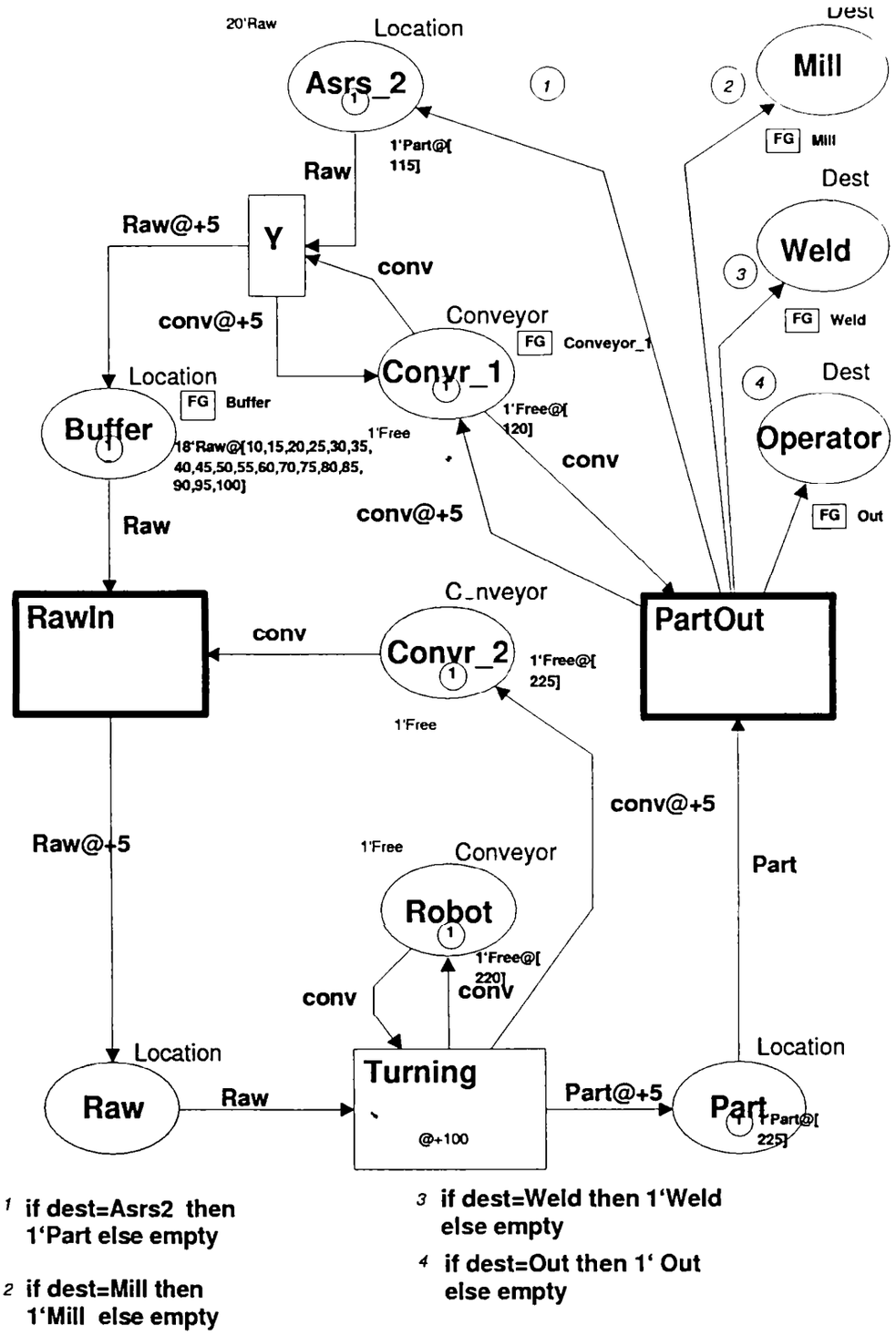


Figura 1.8-1

Exemplele care vor urma vor ilustra aceste situații particulare. Astfel, în Figura 1.8-1 se prezintă cazul în care dintr-un complex de condiții reale din sistem apare situația ca două procese să fie declanşate simultan, adică atât intrarea cât şi ieşirea unei piese din zona de lucru a unui centru de prelucrare. Atât timp cât ambele procese folosesc aceeaşi resursă, de exemplu un conveior, această situație va genera un conflict de resurse. Arbitrarea acestui conflict se poate face pe baza unui criteriu foarte simplu şi anume *"primul proces declanşat este cel care*

duce la scoaterea pieselor din sistem". Cum anume se implementează în sistem această regulă de "comportament" este sarcina acestei lucrări să demonstreze. Figura care ilustrează starea de conflict al resurselor a fost preluată dintr-un model simbolic al unui sistem de fabricație elaborat cu ajutorul rețelelor Petri [Dreucean, M., 1996]. Conveiorul care apare în figură sub denumirea de "Conv_1" este folosit atât pentru încărcare cât şi pentru descărcarea centrului de prelucrare prin frezare, precum şi de alte componente ale sistemului.

Ținând cont că cele două procese "RawIn" şi "PartOut" au condiții de declanşare simultană, lucru imposibil în termeni practici, se impune rezolvarea următoarei dileme: care este modul rațional de desfăşurare a procesului în continuare? Care va fi deci prima tranziție care se va declanşa?

O situație nedorită în funcționarea unui sistem automat de fabricație este atunci când două sau mai multe componente ale sistemului aşteaptă indefinit eliberarea unei resurse ocupată de o altă componentă a sistemului. Această aşteptare întrerupe fluxul de materiale al sistemului automat şi duce astfel la apariția unui blocaj. Pentru multe sisteme automate ieşirea dintr-o stare de blocaj se poate face doar manual, prin eliminarea din sistem a reperelor care au blocat resursa partajată şi reluarea ciclului de fabricație din starea inițială. Un model simplu pentru evidențierea condițiilor de apariție a blocajului este redat în Figura 1.8-2 [vezi Viswanadham, N., Narahari, Y., 1992].

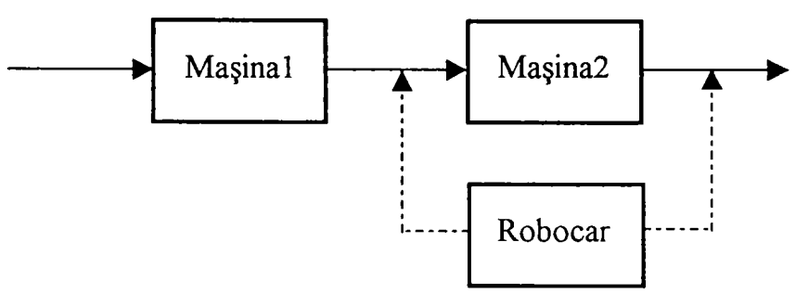


Figura 1.8-2

Robocarul din figură deserveşte maşina numărul 2, asigurând transferul pieselor de la maşina 1 precum şi evacuarea pieselor de pe maşina 2. Dacă robocarul, care are o singură poziție de încărcare, conține o piesă de la prima maşină, în speță piesa 2, în timp ce a doua maşină prelucrează o altă piesă încărcată anterior, în speță piesa 1, atunci descărcarea maşinii 2 nu se mai poate produce şi întregul sistem se va bloca prin întreruperea fluxului de materiale. Această situație este redată în ciclograma din Figura 1.8-3 prin linia verticală întreruptă dublă.

Condițiile practice care favorizează apariția blocajelor într-un sistem de fabricație automat pot fi formulate astfel:

- *excluderea mutuală*, care provine din imposibilitatea folosirii simultane a aceleiaşi resurse de către două unităţi distincte din sistem;
- *blocarea resursei* la nivelul unui proces care nu o eliberează până ce procesul nu se sfârşeşte;
- *blocarea în aşteptare* a unui proces cu o resursă alocată pentru a primi alte resurse care sunt blocate de alte procese;
- *aşteptarea circulantă* între mai multe procese care se aşteaptă în cascadă pentru eliberarea uneia sau a mai multor resurse partajate.

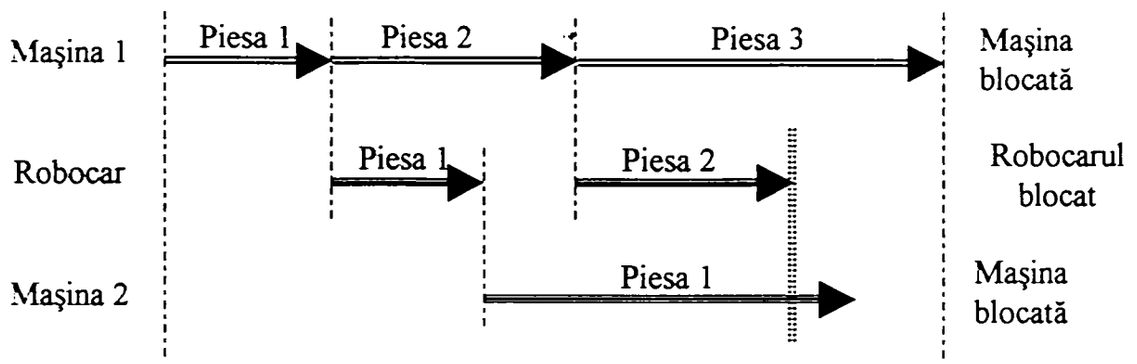


Figura 1.8-3

Toate aceste condiţii favorizante ale apariţiei blocajelor pot fi identificate şi pe schema şi ciclograma din Figura 1.8-2 şi Figura 1.8-3.

Atitudinea proiectantului de sisteme de fabricaţie şi a operatorului care le exploatează faţă de apariţia blocajelor poate îmbrăca următoarele forme:

- *detectarea* blocajelor;
- *resetarea* sistemului din starea de blocaj;
- *prevenirea* apariţiei blocajelor;
- *evitarea* apariţiei blocajelor.

Având în vedere că apariţia blocajelor într-un sistem automat are consecinţe nefavorabile asupra productivităţii sistemului şi asupra indicelui de utilizare a resurselor, problema evitării apariţiei acestora devine de o maximă importanţă. In literatura de specialitate sunt prezentate multe procedee de analiză a sistemelor automate şi de proiectare a sistemelor de control în scopul evitării blocajelor [Viswanadham, N., Narahari, Y., 1992, Zhou, MengChou, 1995, Hrúz, B., 1995]. In esenţă toate aceste procedee se bazează pe controlul stării momentane a sistemului şi pe determinarea în timp real a evoluţiei viitoare a acestuia în scopul evitării blocajelor. *Cum se poate determina cu anticipaţie apariţia condiţiilor de blocaj*

ale sistemului şi alegerea unor rute care să evite blocajele este o altă sarcină asumată de prezenta lucrare.

1.9. Probleme specifice sistemelor flexibile de fabricaţie robotizate. Studii de caz.

1.9.a Generalităţi

Paragrafele care urmează vor prezenta câteva realizări remarcabile ale unor firme europene, care vin să ilustreze noul concept de organizare sistemică a producţiei, alături de noile tehnologii de fabricaţie flexibilă automatizată. Un element important în noul peisaj de fabricaţie începe să fie *robotul industrial*, care preia o parte din sarcinile de manipulare a materialelor în interiorul celulelor de fabricaţie, aducându-şi aportul său la definirea atributului de flexibilitate al acestora [Hans, B., Kief, Olling, G., Waters, T., F., 1986].

Principala funcţie pe care o are un robot industrial într-un sistem de fabricaţie este *funcţia de manipulare*. Un studiu publicat de American Society for Manufacturing Engineers (ASME 1977) arată că la orizontul anilor '90 peste 20% din costurile de fabricaţie ale centrelor de prelucrare vor fi folosite pentru execuţia sistemelor de alimentare şi evacuare, având în vedere că aceste maşini vor trebui să funcţioneze în absenţa unui operator uman.

La fel de importantă ca şi prima aplicaţie este şi utilizarea roboţilor pentru *efectuarea unor sarcini de prelucrare*, prin echiparea trenului condus al robotului cu o unitate de lucru de tipul unităţilor de găurire, vopsire, nituire, debavurare şi multe altele. In acest fel robotul devine un subsistem de prelucrare echipat cu posibilităţi de comandă numerică.

Roboţii industriali usnt foarte potriviţi pentru *desfăşurarea unor lucrări în medii explosive sau în orice fel nocive pentru un operator uman*. Pe lângă nocivitatea chimică se pot include în această categorie şi toate mediile neergonomice care presupun un nivel ridicat de zgomote sau vibraţii, condiţii de muncă stresante prin ritm sau implicaţii emoţionale, etc.

Una din utilizările de mare răspundere ale roboţilor industruiali o constituie *măsurarea robotizată*, parte a controlului automat al calităţii. Roboţii incluşi în această categorie specială de aplicaţii se caracterizează printr-o precizie ridicată, care trebuie să depăşească nivelul de precizie al măsurătorilor efectuate, precum şi printr-o construcţie de mare rigiditate, care sa evite introducerea unor erori de măsurare datorate elasticităţii sistemului. De asemenea echipamentul de control al acestor roboţi impune pretenţii mai ridicate de nivel software.

1.9.b Sistemul Wartsila

Întreprinderea Wartsila Vaasa din Finlanda are în funcțiune un sistem de fabricație flexibilă compus din trei centre de prelucrare identice (CP), un robot de debavurare, o magazie automată, o stație de încărcare-descărcare, o zonă de asamblare preliminară şi un mini computer pentru controlul fluxului de materiale. Sistemul este descris în [Warnecke H.-J., Steinhilper R., 1990]. In "layout"-ul prezentat în Figura 1.9-1 este prevăzut şi un spațiu de rezervă pentru alte centre de prelucrare sau pentru o stație de spălare. Sistemul asigură în fapt prelucrarea unui număr de 150 de repere prismatice diferite, fără nici o oprire sau reglare a maşinilor. Piesele cele mai reprezentative care au fost prelucrate sunt patru tipuri de chiulase de motoare diesel grele şi două tipuri de biele pentru acelaşi gen de motoare. Ambele tipuri de piese se înscriu într-un cub de latură 700 mm şi necesită două sau trei prinderi pentru o prelucrare completă. Timpul de lucru pentru fiecare prindere este între 30 de minute şi două ore.

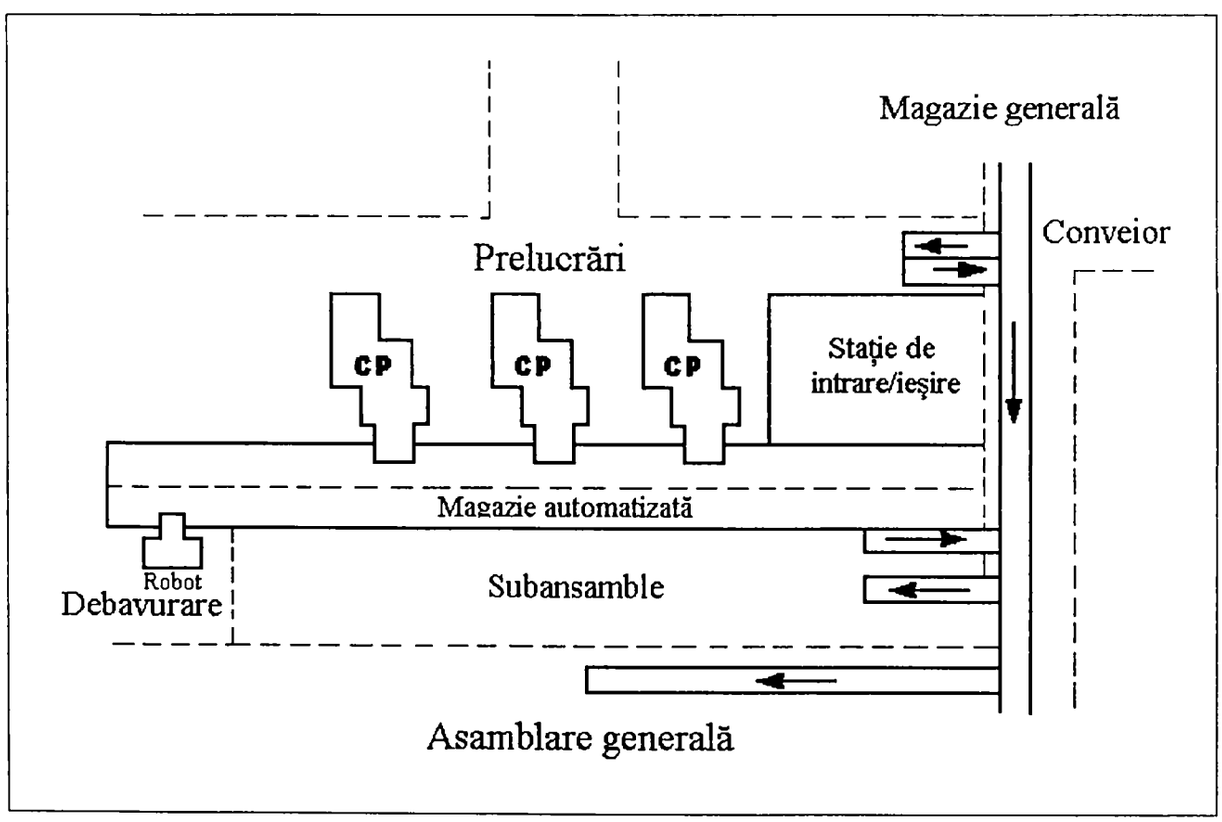


Figura 1.9-1

Sistemul de scule aşchietoare este compus din 200 de scule diferite şi alte 60 de scule derivate. Cel mai mare diametru de sculă este 400 mm iar scula cea mai lungă față de planul frontal al arborelui principal este de 500 mm. Cel mai mare filet prelucrat este M52 şi cea mai

mică gaură este de 5 mm. *Precizia de prelucrare* minimă este exprimată prin clasa de precizie IT6 la găurire, 0,02 mm precizie de prelucrare la frezare şi rugozitatea minimă de R_a 1,6 μm.

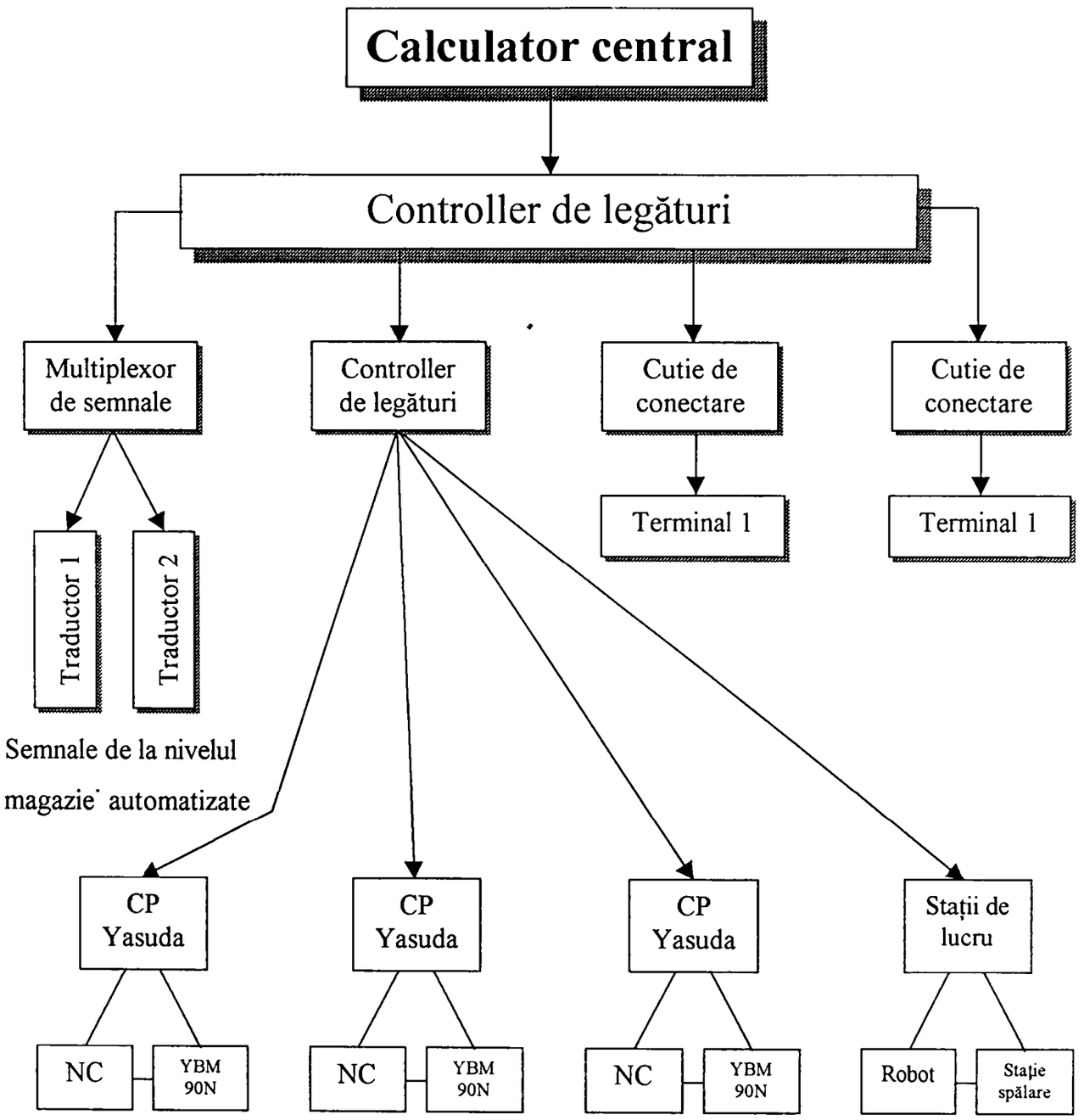


Figura 1.9-2

In centrul sistemului FMS de la Wartsila se află o *magazie automatizată*, deservită de un translator de magazie cu două axe, produs de o firmă suedeză. In această magazie se află stocate diferite materiale, cum ar fi semifabricate, piese finite, diverse subansamble, dispozitive de prindere, etc. Toate aceste elemente diferite sunt aăezate pe palete identice, astfel încât manevrarea lor de către translatorul de magazie să se facă în mod unitar. Semifabricatele se instalează pe palete pe nişte carucioare cu roti, care sunt apoi aduse în dreptul translatorului

pentru a permite preluarea paletelor şi stocarea lor în magazie. De aici semifabricatele şi toate celelalte elemente depozitate sunt livrate automat conform unui program de fabricaţie ce rulează pe un calculator principal. De regulă protocolul de transfer în magazie este FIFO (first-in- first- out)

Centrele de prelucrare sunt de tipul YMB-90 Ns, produse de firma Yasuda din Japonia. Magaziile de scule sunt în număr de 5, fiecare cu câte 60 de scule. Sistemul de schimbare a sculelor are două braţe şi poartă sculele prin faţa unui sistem de curăţire cu perie rotativă şi jet de aer, atât la prinderea cât şi la desprinderea sculelor din arborele principal al centrului de prelucrare.

Sistemul de diagnosticare al centrului de prelucrare permite operatorului să monitorizeze în permanenţă funcţionarea maşinii şi să primească informaţii foarte detaliate în caz de avarie. Există de asemenea un sistem de protecţie la ruperea sculei.

Schimbătorul de palete al centrului de prelucrare are două poziţii, fiecare paletă care se instalează pe masa de lucru fiind identificată individual pe baza unui cod magnetic analizat cu un cititor special.

Sistemul de comandă şi control este alcătuit dintr-un calculator central, legat de echipamentele periferice prin intermediul unei cutii de conexiuni. Periferia este alcătuită din două terminale de comunicaţie, controlerele centrelor de prelucrare şi ale staţiilor de lucru, precum şi sistemul de senzori al magaziei automatizate (vezi Figura 1.9-2).

Memoria sistemelor CNC din compunerea centrelor de prelucrare conţine toate programele de lucru pentru reperele specifice sistemului, astfel încât lansarea în execuţie a unui anumit reper nu presupune nici un fel de operaţii suplimentare de punere la punct sau de reglaj. Reperul este recunoscut prin intermediul paletei pe care este montat şi programul de prelucrare adecvat este automat lansat în execuţie, după verificarea magaziei de scule care trebuie să conţină toate poziţiile necesare rulării programului.

Asamblarea reperelor şi controlul dimensional se fac manual, cu un număr de 8-10 muncitori pe schimb. Această modalitate de organizare a fost special adoptată pentru a "umaniza" într-o anumită măsură sistemul flexibil de fabricaţie şi de a crea operatorilor umani senzaţia de utilitate şi de importanţă a muncii depuse, tot mai necesară pe măsura dezvoltării sistemelor de fabricaţie flexibilă care conduc spre perioada "post-CIM". Din acelaşi motiv nu există nici sisteme de control automat, problema preciziei fiind lăsată în seama operatorilor de la asamblare care dispun de toate dispozitivele de verificare necesare.

1.9.c Sistemul de fabricație flexibilă de la uzinele Citroën

Fabrica de la Meudon, lângă Paris, aparținând concernului Citroën, execută prototipuri pentru toate celelalte firme ale concernului, precum şi pentru alte firme partenere angajate în cursele de raliuri şi de Formula I. Sistemul conține doar două centre de prelucrare, (vezi Figura 1.9-3) cu ajutorul cărora se pot prelucra orice fel de piese turnate aflate în specificul de fabricație al firmelor respective. Ciclul de producție maxim este de 5 zile pentru orice reper. Deoarece varietatea de repere este foarte mare, partea principală a sistemului flexibil prezentat este partea de CAD/CAM, unde sunt elaborate programele de prelucrare şi tehnologiile de execuție pentru aceste repere.

Sistemul de transfer este constituit din mai multe AGV- uri care se deplasează pe traiectorii marcate inductiv în pardoseală. Acestea transferă materiale diverse, de la semifabricate la scule sau dispozitive de fixare.

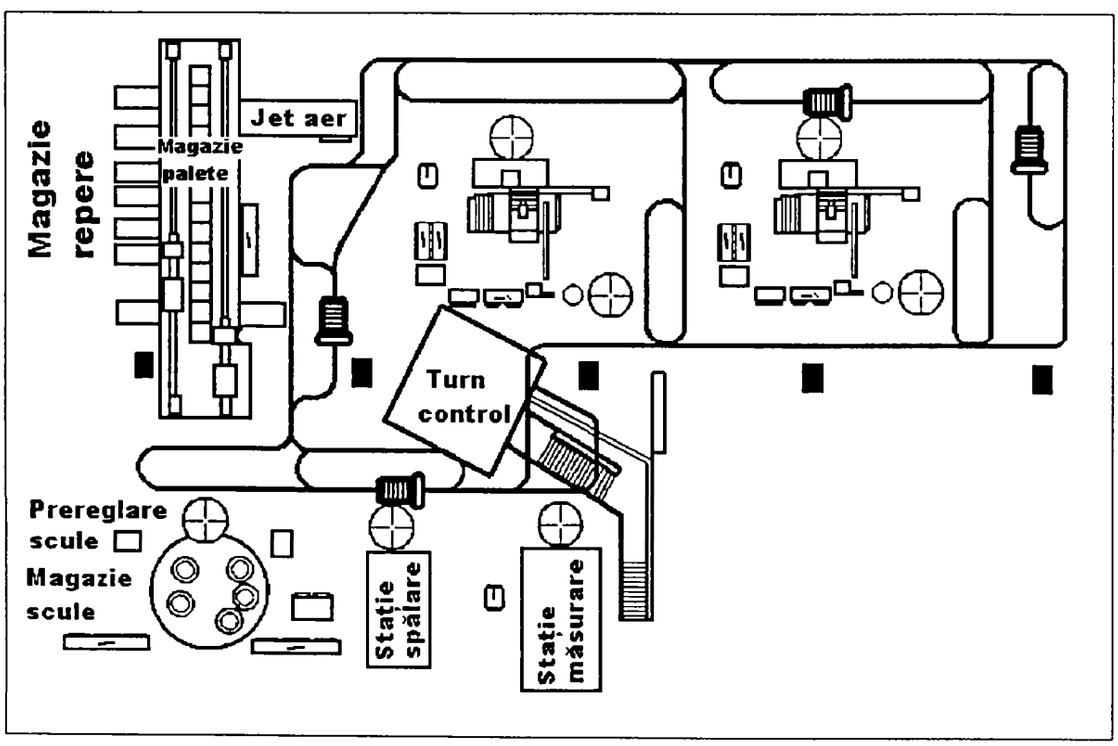


Figura 1.9-3

Prelucrările mecanice sunt executate pe două *centre de prelucrare* de tip Graffenstaden cu 5 axe, fiecare fiind echipat cu două magazii de scule cu lanț cu 50 de poziții (CP1, CP2 în Figura 1.9-3). Spațiul de lucru este un cub cu latura de 500 mm, având ca bază suprafața unei palete patrate cu latura de 800 mm. Intre magazia centrală de scule şi fiecare centru de prelucrare au fost configurate trasee de transfer deservite de AGV, care să asigure alimentarea cu scule noi sau cu seturi de scule specifice unor anumite prelucrări. Acest lucru se

realizează prin încărcarea cu un manipulator a unui număr de 20 de scule din magazia centrală care are un total de 600 de poziții, pe un tambur dispus pe AGV.

AGV-ul transferă tamburul cu sculele la centrul de prelucrare, unde se găseşte un tambur identic, conținând sculele ce urmează să se întoarcă în magazia centrală din motive diverse: depăşirea duratei de aşchiere programate, ruperi accidentale, nevoia de a elibera pozițiile ocupate în lanțul magaziei centrului de prelucrare, etc. Un manipulator cu braț rotativ cu două poziții schimbă între ele sculele din cele două tambururi, astfel încât AGV-ul va transfera înapoi tamburul cu sculele disponibilizate de către centrul de prelucrare, în timp ce sculele aduse recent vor fi instalate în lanțul magaziei de scule a centrului de prelucrare pentru a putea fi utilizate de către acesta. Sistemul central de comandă ține gestiunea sculelor, având sarcina de a actualiza în permanență starea acestora din punct de vedere al duratei de viață şi al poziției în sistem. In acest mod sculele pot fi instalate pe oricare din cele două centre de prelucrare, fără nici un fel de restricții, pregătirea şi înmagazinarea lor făcându-se centralizat. *Pregătirea sculelor* se face de către un singur angajat pe fiecare schimb, acesta având la dispoziție o maşină de prereglare a sculelor legată la calculatorul central care preia astfel automat toate datele despre sculă.

1.9.d Sistemul Cessna

Sistemul Cessna a fost realizat în Marea Britanie, în Glenrothes, la o fabrică producătoare de echipamente hidraulice. Pe acest sistem a fost calată producția de pompe cu roți dințate, care implică o varietate de circa 1000 de modele distincte, deşi numărul de repere din compunerea unei astfel de pompe este relativ redus (5-6 repere). Sistemul a fost conceput în jurul anului 1983 şi a fost finalizat în circa doi ani prin aportul de echipamente şi tehnologie al firmei British Olivetti.

"Layout"-ul sistemului (vezi Figura 1.9-4) relevă că sistemul este alcătuit dintr-un număr de 5 conveioare de intrare-ieşire, fiecare fiind destinat să transfere unul din reperele care alcătuiesc pompa cu roți dințate care se prelucrează în sistem. Capacitatea de stocare a acestor conveyoare, folosite şi ca magazii intermediare, este de patru ore de producție. Fiecare conveior are două nivele. Pe cel superior sunt aşezate manual semifabricatele, iar pe cel inferior sunt depuse piesele finite de către robotul ce deserveşte celula de fabricație.

Componenta principală a celulei este un *robot pe şine*, de tip Olivetti, care este echipat cu un gen de tavă în care sunt aşezate reperele din compunerea produsului finit. Acest robot

deserveşte cele trei centre de prelucrare identice, stația de spălare a pieselor, cele cinci conveioare, maşina de măsurare în coordonate şi celula pentru operații periferice, de regulă executate manual (debavurări, asamblări parțiale, verificări, etc.).

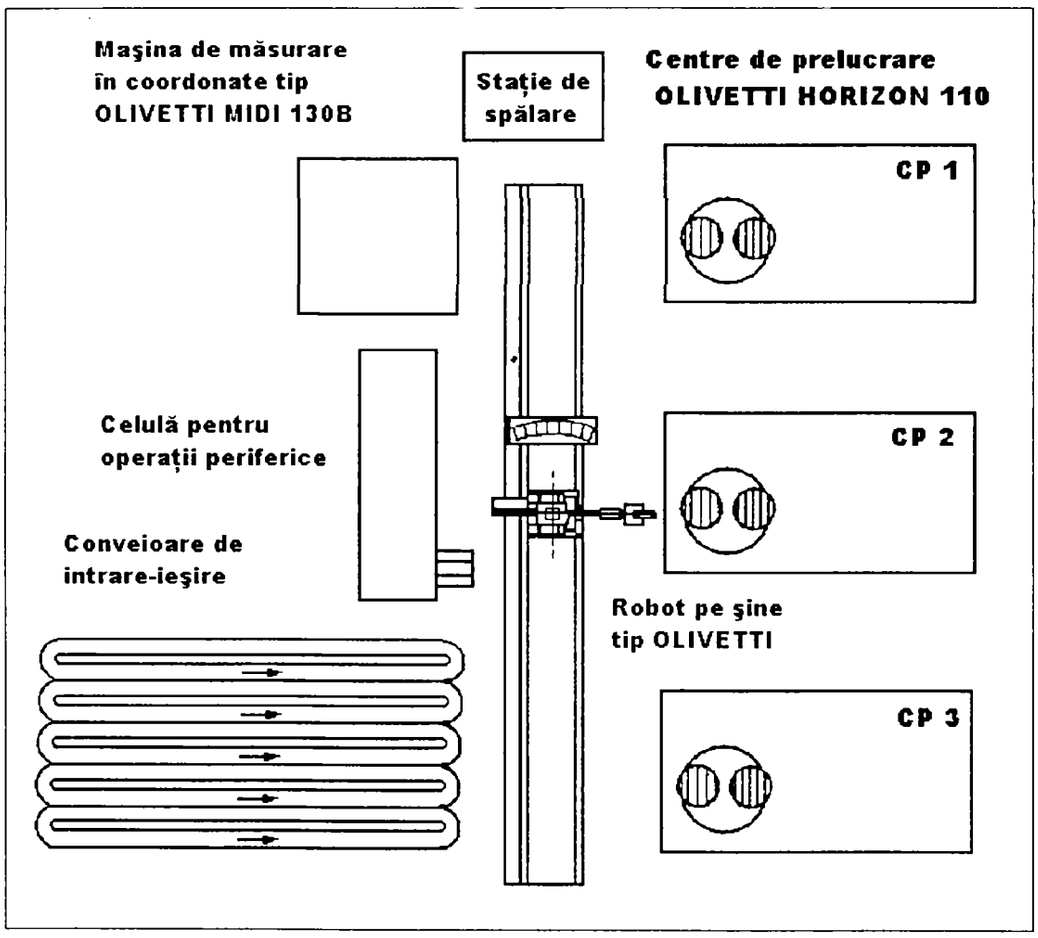


Figura 1.9-4

Centrele de prelucrare, de tip Olivetti Horizon 110B, sunt prevăzute cu mese schimbabile cu două poziții, pe care sunt aşezate paletele cu repere. Paletele au fost proiectate astfel încât permit fixarea simultană a tuturor celor cinci repere care se prelucrează în sistem, astfel încât toate paletele vor fi identice. Fixarea reperelor se face automat, cu sisteme de prindere acționate mecanic sau pneumatic.

După prelucrare, reperele sunt curățate de aşchii şi sunt uscate prin suflare cu aer cald, după care sunt transferate la maşina de măsurare. Aici este verificat un set de cote considerate de importanță primordială pentru buna funcționare a pompei, precum şi pentru precizia de prelucrare a maşinii. Dacă la anumite reper şi cote sunt atinse limitele câmpului de toleranță, sistemul de control al celulei sesizează centrul de prelucrare de unde provine reperul respectiv pentru a interveni prin schimbarea sculei care poate fi deja uzată, sau prin aplicarea unor corecții în programul NC de prelucrare.

Sistemul este condus de un calculator central, care are şi rol de supervizor pentru echipamentele NC ale centrelor de prelucrare. Astfel, dacă un asemenea echipament eşuează, computerul central poate prelua funcţiile acestuia continuând astfel activitatea în celulă. Ciclul de lucru este organizat pentru prelucrarea unor loturi de 50-500 de repere identice, nefiind posibilă, sau fiind prea puţin productivă lansarea în execuţie a reperelor la întâmplare. Sistemul presupune existenţa unui număr de 2-3 operatori umani de pregătire profesională destul de ridicată, care să preia sarcinile de alimentare periodică cu semifabricate şi pe cele de reglaj a centrelor de prelucrare precum şi de pregătire a sculelor.

1.9.e Sistemul Kloeckner-Humboldt-Deutz

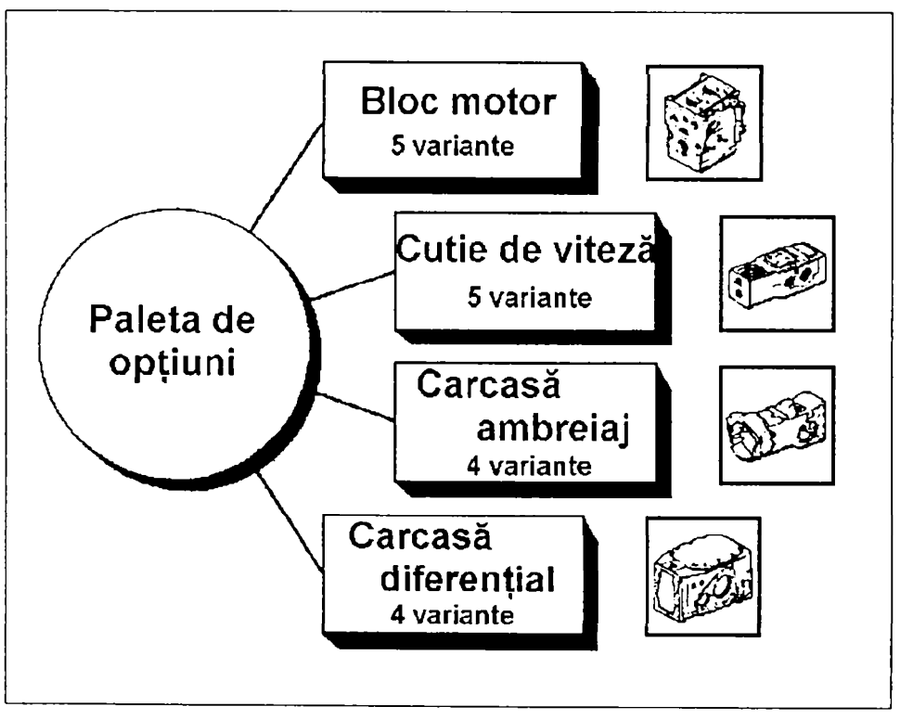


Figura 1.9-5

Industria anilor '80 în RFG a fost pentru prima oară pusă în faţa unor cerinţe ale pieţei greu de satisfăcut cu dotările şi cu mentalitatea tehnică a acelor ani. Problema care trebuia rezolvată era dacă cerinţele tot mai diverse de pe piaţă, care fărâmiţau practic loturile de fabricaţie din ce în ce mai mult, trebuiau întâmpinate cu noi linii de flux tehnologic amenajate pe baza conceptului tehnologic de "tehnologie de grup" sau trebuia găsit altceva, un nou concept, care să elimine investiţia imensă presupusă de liniile tehnologice, fără şanse de amortizare în cazul loturilor mici de fabricaţie.

Analizându-se în mod serios spectrul de produse la care variabilitatea este maximă, (vezi Figura 1.9-5), s-a ajuns la concluzia că problema poate fi rezolvată cel mai economic prin

amenajarea unui sistem flexibil. Firma constructoare a sistemului a fost Burkhardt & Weber. (vezi Figura 1.9-6)

Sistemul are în componență 4 *centre de prelucrare* identice, de tip MC100, care se pot înlocui complet unul pe altul. Aceste centre de prelucrare sunt adecvate pentru prelucrarea unor repere de tip carcasă. Maşinile au axa orizontală, iar din necesitatea de a mări viteza de execuție a reperelor cu multe găuri, la primul centru de prelucrare s-a prevăzut un sistem de găurire cu capete multiax, care se pot încărca după nevoi dintr-o magazie specială, ataşată acestei maşini. Sistemul este deservit din punct de vedere al transferului de un conveior pe şine. *Incărcarea paletelor* cu repere se face manual la stațiile de încărcare.

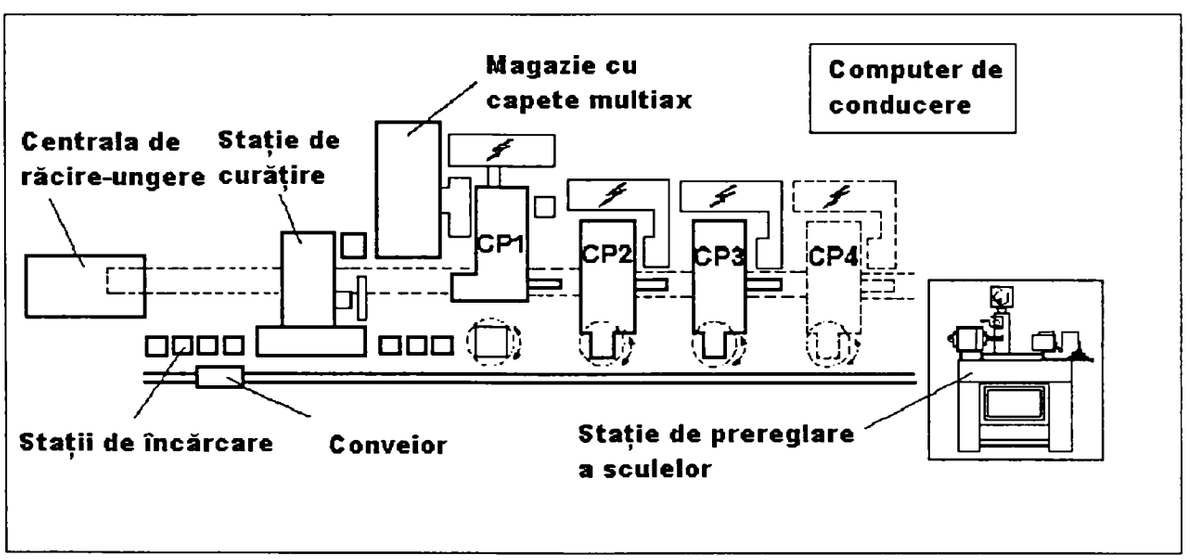


Figura 1.9-6

Fiecare stație din sistem este obiectul unei *activități de control organizată ierarhic* (vezi Figura 1.9-7). Creierul întregii activități este un computer de conducere, integrat prin intermediul unei rețele locale cu celelalte echipamente sau stații de lucru. Sarcinile pe care le îndeplineşte fiecare nivel al ierarhiei sistemului de conducere pot fi identificate în figură. Este de remarcat faptul că sistemul de transfer are un controller separat, lucru care demonstrează importanța acestui echipament pentru funcționarea în siguranță şi cu mare eficiență a întregii instalații.

Pe primul nivel de control sunt rezolvate următoarele sarcini:

- gestiunea generală a fişierelor;
- încărcarea maşinilor;
- gestiunea sculelor;
- lansarea loturilor de producție.

Pe nivelul doi sunt grupate următoarele obiective:

- controlul fluxurilor de date;
- distribuția programelor NC;
- efectuarea de calcule concrete;
- rularea programelor NC;
- monitorizarea uzurii sculelor;
- manipularea materialelor.

Pe ultimul nivel se rezolvă doar transferul pieselor, ca o componentă esențială a sistemului de conducere. La acest nivel este instalat un calculator de tipul Simatic S5 cu toată periferia necesară.

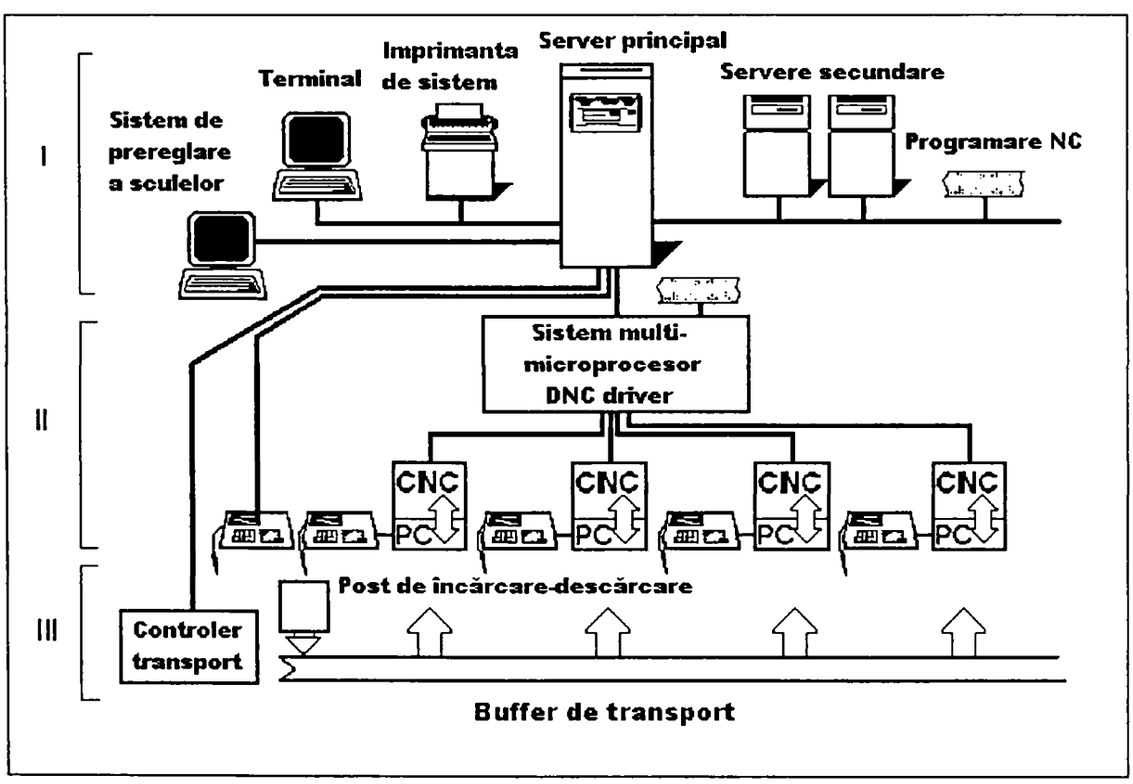


Figura 1.9-7

Calculatorul principal de comandă este de tipul Siemens 300 şi el are sub control toate echipamentele de conducere instalate în sistemul flexibil. *Programarea producției* se face manual, pe baza comenzilor şi a strategiei generale a întreprinderii, computerul principal de comandă având doar misiunea de a urmări şi de a asigura îndeplinirea în bune condiții a planului astfel elaborat.

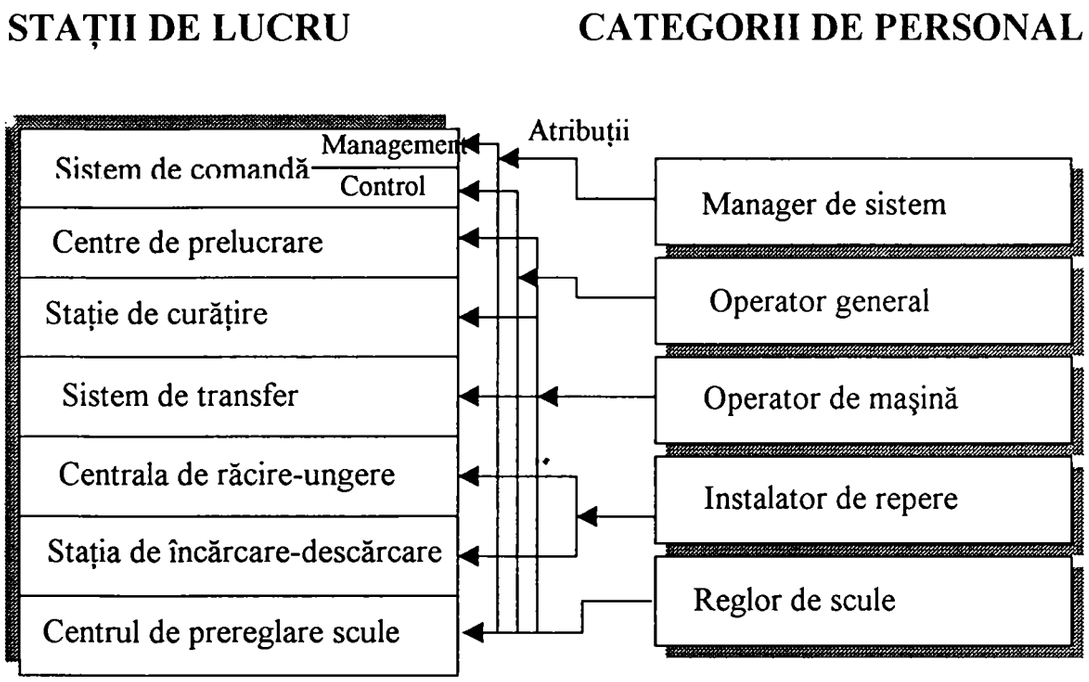


Figura 1.9-8

Una din dimensiunile principale a implementării unui sistem flexibil de fabricație este *dimensiunea socio-umană.* In cazul exemplului prezentat, componenta umană a necesităților de operare din sistem este destul de importantă, ținând cont că însăşi programarea producției se face manual, la aceasta adăugându-se instalarea semifabricatelor pe palete, reglarea maşinilor, prereglarea sculelor, descărcarea pieselor şi multe altele. Deci operatorul uman este nevoit să coabiteze cu sistemul şi să se adapteze la un nou stil de muncă, în care elementul inedit este prezența maşinilor în ciclu automat, care impun un ritm obiectiv şi uneori stresant. Pornind de la această observație, conducerea companiei a elaborat un studiu de integrare a forței de muncă în sistem, începând cu dotarea cu personal uman şi ajungând la distribuirea de sarcini pe principalele stații de lucru (vezi Figura 1.9-8).

1.10. Breviar bibliografic

In cele ce urmează este prezentat un breviar bibliografic extras dintr-o bază de date pe care autorul a alcătuit-o în etapa de documentare pentru prezenta lucrare de doctorat. Sunt cuprinse principalele titluri care au stat la baza formării fondului teoretic al prezentei lucrări, multe dintre ele fiind referite direct în lucrarea de față. Câmpul "Rezumat" al bazei de date cuprinde principalele subiecte abordate în fiecare lucrare care se asociază tematic cu titlul tezei de doctorat.

Date primare

14-Apr-99

Nr. 12
Titlul: MADS-Wizzard. Manual de utilizare
Autor: ***
Anul apariției: 1987
Rezumat: Manual de utilizare pentru un nucleu de timp real multi-tasking

Nr. 2
Titlul: Object-Oriented Modeling and Design
Autor: James Rumbaugh, Michael Blaha, William Premerlani,
Anul apariției: 1991
Rezumat: Introducere.
Modelarea ca tehnica de proiectare.
Modelarea obiectelor.
Concepte avansate de modelare a obiectelor.
Modelarea dinamica.
Modelarea funcționala.
Analiza sistemelor.
Proiectare unui sistem.
Proiectarea obiectelor.
Implementarea modelelor.
Limbaje orientate spre obiecte.
Limbaje neorientate spre obiecte.
Baze de date relaționale.
Compilator pentru diagramele cu obiecte.
Animația pe computer.
Proiectarea unui sistem de distribuție a energiei electrice.

Nr. 3
Titlul: The Design and Operation of FMS
Autor: Dr. Paul Ranky

Anul apariției: 1983

Rezumat: Introducere in teoria FMS
Managementul in proiectarea si implementarea unui FMS
Distributed processing in FMS
Integrated CAD/CAM systems & part programing
Baze de date distribuite pentru scule si dispozitive
Utilizarea roboților industriali, a magaziilor automate si ale AGV-urilor
Maşini de măsurare in coordonate
Interfațarea computerelor cu celelalte componente ale FMS
CAPP si programul de producție
Considerații economice
Aspecte socio-umane

Nr. 4

Titlul: Flexible Manufacturing Systems

Autor: H.-J. Warnecke R. Steinhilper

Anul apariției: >1989

Rezumat: Planificarea si organizarea producției in FMS.
Asupra proiectării sistemelor flexibile- orientarea spre produs.
Arhitectura sistemelor de control a FMS.
Controlul automat in FMS.
Aplicații ale roboților in FMS.
Dezvoltarea maşinilor unelte si evoluția lichidelor de răcire.
Studii de caz.

Nr. 5

Titlul: Japan - USA Symposium on Flexibel Automation

Autor: *****

Anul apariției: JULY 14-18, 1986

Rezumat: 1. A solution of Inverse Dynamics of Manipulators with Elasticity and Backlashes at Joints
2. A Trajectory Generation Algorithm for Straight Line Motion in cartezian Space
3. A Sliding Mode Intelligent Servo System to Improve the Path Accuracy and Power Consumption of Robot Manipulators
4. Optimization Principles for Computer Integrated Manufacturing Systems
5. Flexible Automated Production System for Automotive Radiators
6. The Design of Expert Systems for Planning Flexible Manufacturing
7. An Expert System for Automatic Process Planning of Boring Operation by PROLOG
8. An Inference Engine for Real-Time Fuzzy Control

Nr. 6

Titlul: Performance Modeling of Automated Manufacturing Systems

Autor: N. Viswanadham Y. Narahari

Anul aparitiei: 1992

Rezumat: Sisteme de producție automate
-sisteme de producție
-măsurarea performanțelor
-maşini conduse prin calculator
-sisteme de manevrare a materialelor
-lay-out-ul unei secții de producție
-sisteme de fabricație flexibila
-sisteme de control prin calculator
Modelarea cu lanțuri Markov
Modelarea cu şiruri de aşteptare
Modelarea cu rețele Petri

Nr. 7

Titlul: Concurrent Engineering

Autor: Hamid R. Parsaei, William G. Sullivan

Anul aparitiei: 1993

Rezumat: Elemente de organizare in mediul concurent
-principiile ingineriei concurente
-implementarea sistemelor concurente
-imbunătățirea comunicării interpersonale in cadrul sistemelor concurente
Tehnici si metode ale ingineriei concurente
-modele ale procesului de proiectare
-proiectarea concurentă pe bază de decizie
-optimizarea concurentă a procesului de proiectare si prelucrare
-sisteme concurente conduse prin calculator
-optimizarea proiectării multiatribut
-proiectarea concurenta a celulei de fabricație si a sistemului de control
-metodologie generală de evaluare a prelucrabilității
-optimizarea concurenta a erorilor de proiectare si de fabricație
Considerente de cost in ingineria concurentă
-proiectarea orientata spre costuri
-priectarea economică in mediu concurent
Inteligenta artificiala in ingineria concurenta
-aplicații ale sistemelor expert in proiectare
-utilizarea bazelor de cunoştințe la proiectarea orientata spre prelucrare
folosind trasaturi esențiale
-acumularea concurenta de cunoştințe
-modelarea procesului de proiectare cu rețele Petri

Nr. 8
Titlul: Design, Analysis, and Control of Manufacturing Cells
Autor: ***
Anul aparitiei: 1991
Rezumat: Controlul parametrilor fizici ai unor procese de prelucrare intr-o arhitectura
deschisa;
Controlabilitatea sistemelor flexibile;
Sistem de proiectare a celulelor autonome de prelucrare cu retele Petri;
Arhitectura soft a unei celule de fabricatie independente;
Proiectarea celulelor independente-Traficul intercelular;
Proiectarea si implementarea algoritmica a unui controler in timp real pentru o celula de fabricatie;
Reducerea timpului de lucru cu ajutorul sistemelor celulare interconectate;
Decuploare pentru o celula flexibila de extrudare.

Nr. 9
Titlul: Monitoring and Control for Manufacturing Processes
Autor: ***
Anul aparitiei: 1990
Rezumat: Monitorizarea conditiilor de prelucrare in cazul automatizării cu retele neuronale;
Recunoaşterea on-line a uzurii burghiului cu ajutorul retelelor neuronale;
Aplicatii ale retelelor neuronale la probleme de control automat al calitătii;
Sistem de control pe doua nivele a fluxului de informatie intr-un sistem flexibil;

Nr. 1
Titlul: Artificial Inteligence, Simulation and Modeling
Autor: Lawrence E. Widman, Keneth A. Lopato, Norman R.
Anul aparitiei: 1989
Rezumat: 1.Issues in Qualitative Reasoning about Diffusional Processes. -Shoshana L. Hardt
2.A Hybrid Paradigm for Modeling ofComplex Systems. -Sergio Ruiz-Mier, Joseph Talavage
3.The Role of Artificial Intelligence in Discrete-Event Simulation. -Robert M., O'Keefe

Nr. 11
Titlul: Working with Neural Networks
Autor: Dan Hammerstrom

Anul apariției: 1992

Rezumat: Prezentare generala a domeniului rețelelor neuronale, definiții de termeni, algoritmi principali

Nr. 22

Titlul: Petri Nets in Flexible and Agile Automation (collection of articles)

Autor: MengChu Zhou

Anul apariției: 1995

Rezumat: Introduction to Petri Nets in Flexible and Agile Automation. 1
Application of Petri Nets to Sequence Control Programming. 43
CAD of Logic Controllers with Petri Nets. 71
Automatic Generation of Sequence Control Programs via Petri Nets and Logic Tables for Industrial Applications. 93
Planning and Scheduling based on Petri Nets. 109
Petri Net-Based Heuristic Scheduling for Flexible Manufacturing. 149
Scheduling and Rescheduling of AGVs for Flexible and Agile Manufacturing. 189
Stochastic Petri Net Models of Communication and Flexible Systems. 207
Deadlock Avoidance Policy for Flexible Manufacturing Systems. 239
Discrete-Event Control Design for Manufacturing Systems via Ladder Logic Diagrams and Petri Nets: A Comparative Study. 265
From State Transition Models to DFD Extended Methods for Specifying Reactive Systems. 305
Supervisory Control Specification and Synthesis. 337

Nr. 13

Titlul: Conducerea automata a proceselor

Autor: Ion Babutia,Toma Leonida Dragomir, Ioan Muresan,

Anul apariției: 1985

Rezumat: Aspecte privind utilizarea sistemelor multimicroprocesor in conducerea
automata a sistemelor pag 124-198;
Utilizarea SMM pentru conducerea automata in timp real a proceselor industriale pag 201-222

Nr. 14

Titlul: Automatica Management Calculatoare

Autor: ***

Anul apariției: 1983

Rezumat: Filtrajul optimal numeric Kalman-Bucy in rezolvarea problemelor procesuale

pag 62-87;
Programarea in mediu concurent pag 121-142

Nr. 15
Titlul: Automatica Management Calculatoare
Autor: ***
Anul aparitiei: 1984
Rezumat: Studiu privind modelele instruibile de decizie in domeniul social pag 7-24;
Un proces de alocare optimala multiobiectiv a resurselor de apa pag 35-47

Nr. 16
Titlul: Automatica Management Calculatoare
Autor: ***
Anul aparitiei: 1984
Rezumat: Optimizarea si analiza sistemelor stocastice cu evenimente discrete cu aplicare in automatizarea fabricației;
Rețele simple si şiruri de aşteptare pag 127-147;

Nr. 17
Titlul: IEEE Transactions in Neural Networks
Autor: ***
Anul aparitiei: 1993
Rezumat: Implementaion of Self-Organizing Neural Networks for Visuo-Motor Control of an Industrial Robot page 86-95

Nr. 18
Titlul: Procese si sisteme informaționale
Autor: ***
Anul aparitiei: 1982
Rezumat: Ingineria concepției: metoda deciziilor descendent structurate pag. 215-227

Nr. 19
Titlul: Systemique-Theorie & applications
Autor: ***
Anul aparitiei: 1992
Rezumat: Generalități, Metodologie, Formalisme matematice, Practica sistemică,
Cercetări sistemice

Nr. 20
Titlul: An Introduction to Neural Networks

Autor: Ben J. A. Krose, P. Patrick van der Smagt

Anul apariției: 1994

Rezumat: Generalități
Teorie (Adaline and Perceptron, Back-Propagation, Self-Organizing
Networks, Recurrent Networks, Reinforcement learning)
Aplicații (Robot control, Vision)
Implementări (General Purpose Hardware, Dedicated Neuro-Hardware)

Nr. 21

Titlul: Flexible Fertigungssysteme

Autor: Gunther Ropohl

Anul apariției: 1971

Rezumat: Producția industrială ca sinteză a tehnicii, tehnologiei si organizării,
Teoria ciberneticii,
Flexibilitatea ca soluție a postulatului automatizării sistemelor de producție,
Flexibilitatea ca semn distinctiv al unei clase de sisteme de producție,
Concepția tehnologic-constructiva a sistemelor de fabricație flexibile.

Nr. 10

Titlul: Computer Integrated Manufacturing Handbook

Autor: Eric Teicholz, Joel n. Orr

Anul apariției: 1987

Rezumat: Planificarea proceselor pag 2.115;
Sisteme de control numeric pag. 2.122;
Robotica pag. 2.149;
Planificarea resurselor materiale si controlul stocurilor pag 2.166;
Planificarea si controlul producției pag 2.216;
Cim si comunicațiile pag2.228;
Rolul manipulărilor de materiale pag. 2.248;

1.11. Bibliografie

1. ***, 1982, Procese si sisteme informaționale, Editura Academiei RSR, Bucureşti.
2. ***, 1992, Systemique-Theorie & applications, Culegere de articole, Technique & Documentation-Lavoisier, Paris.
3. Dreucean, M., 1996, Modeling and Simulation - Tools for Evaluation in Flexible Manufacturing Systems, Pohjois Savon Polytechnic, Kuopio, Finlanda.
4. Gallou, F. le, Buchon-Meunier, B., 1992, (coordonnateurs), Sistemique, Théorie et Application, Technique et documentation-Lavoisier, Paris.
5. Hans, B., Kief, Olling, G., Waters, T., F., 1986, The International CNC Reference Book, Becker Publishing Company (UK) Ltd.
6. Hrúz, B., 1995, Control of Discrete Event Dynamic Systems in the Context with Flexible Manufacturing Systems and Petri Nets, Helsinki University of Technology, Control Engineering Laboratory, Report 100, June 1995.
7. Kovacs, Fr., Grigorescu, S., Rădulescu, C., 1994, Sisteme de fabricație flexibilă robotizate, Părțile I şi II, Litografia UPT, Timişoara.
8. Lawrence E. W., Keneth A. L., Norman R. N.,1989, Artificial Inteligence, Simulation and Modeling, John Wiley & Sons, New York, Chichester, Brisbane, Toronto, Singapore.
9. Ranky, Dr. P., 1990, The Design and Operation of FMS, IFS (Publications) Ltd., UK, North Holland Publishing Company.
10. Ropohl, G., 1971, Flexible Fertigungssysteme, Otto Krausskopf-Verlag GmbH, Mainz;
11. Shannon, G., E., 1948, A Mathematical Theory of Communication, BSTJ 27.
12. Skach, C., 1985, Tehnica prelucrării informațiilor, Curs, ediția I-a, Litografia I.P.Timişoara.
13. VDI - Gemeinschaftsausschuß CIM, 1991, Rechnerintegrierte Konstruktion und Produktion, Band 5: Produktionslogistik, VDI - Verlag GmbH, Düsseldorf.
14. Viswanadham, N., Narahari, Y., 1992, Performance Modeling of Automated Manufacturing Systems, Prentice Hall, Englewood Cliffs, New Jersey.
15. Warnecke H.-J., Steinhilper R., 1990, Flexible Manufacturing Systems, IFS (Publications) Ltd, UK, Springer-Verlag Berlin.
16. Zhou, MengChou, 1995, Petri Nets in Flexible and Agile Automation, Kluwer Academic Publishers, Boston, MA.

2. MODELAREA ŞI SIMULAREA ÎN PROIECTAREA UNUI SISTEM FLEXIBIL DE FABRICAȚIE. CELULE DE FABRICAȚIE INDEPENDENTE

2.1. Cuprins

2.2. Asupra semnificației noțiunilor de "modelare" și "simulare"

In vorbirea curentă noțiunile de "*modelare*" și "*simulare*" au de multe ori semnificații foarte apropiate. O departajare semantică a acestor două noțiuni este utilă cel puțin din punct de vedere al preciziei exprimării.

Prin *"modelare"* se înțelege abstractizarea unor subsisteme sau funcții ale acestora, cu scopul unei mai bune înțelegeri, înainte de a trece la realizarea lor practică [vezi Rumbaugh, J., Blaha, M., ș. a., 1991, pagina 7]. In acțiunea de modelare, *abstractizarea* are un rol deosebit. Această noțiune definește activitatea de orientare asupra aspectelor esențiale, fundamentale, ale unei entități, ignorând proprietățile ei accidentale, nespecifice.

Prin "*simulare*" se înțelege "rularea" modelului, adică transferarea obiectelor în lumea reală. Este evident că pentru a putea desfășura o procedură de simulare este necesar să existe mai întâi un model. Cu aceste precizări, în continuarea lucrării termenii de modelare și simulare vor fi folosiți în mod propriu, deși în literatura de specialitate nu se face în mod sistematic diferențierea necesară.

Așa după cum se arată în [Ruiz-Mier, S., Talavage, J., 1989], diferite metode de simulare au fost introduse începând din anii '60, cu scopul de a face procesul de simulare un proces cât mai natural. Printre aceste procedee, *modelarea proceselor cu evenimente discrete* a devenit foarte obișnuită, pe această bază dezvoltându-se alte metode puternice, cum ar fi *modelarea pe bază de rețele.* Aceste metode bazate pe utilizarea rețelelor oferă modelatorului concepte simple dar puternice pentru a capta aspectele particulare de comportament ale sistemului. Rețelele de șiruri de așteptare, lanțurile Markov sau rețelele Petri sunt câteva exemple de astfel de abordări. Cu toate avantajele oferite de simplitatea în utilizare și de transpunerea corectă a comportamentului sistemului în model, etapa de decizie în dezvoltarea modelului nu poate fi rezolvată. In condiții de concurență și de partajare a unor resurse ale sistemului, apare un *grad de incertitudine în dezvoltarea simulării*, pe care metodele bazate pe rețele nu îl pot rezolva. Exemple de acest gen vor fi prezentate în capitolele următoare ale lucrării.

Pentru a putea completa modelul în partea de luare a deciziilor, este nevoie de folosirea elementelor de *inteligență artificială.* Pe această linie s-au impus două metode de modelare care pot rezolva probleme de incertitudine pe baza unui sistem de evaluare de cunoștințe. Acestea sunt programarea orientată pe obiecte și programarea logică.

Programarea orientată pe obiecte folosește un set de simboluri care marchează obiectele, asociate cu o bază de date unică, care cuprinde proprietățile și operațiile care

definesc obiectul. Baza de cunoştinţe din care fac parte obiectele este organizată pe clase ierarhice. Comunicarea dintre obiectele din clase diferite se face prin transferul unor mesaje. Mesajele de comandă nu spun nimic despre felul cum trebuie îndeplinită o a numită sarcină, ci doar care este aceasta. Această paradigma a transferului de informaţie dintre obiectele aparţinând diferitelor clase conferă modularitate sistemului şi îl apropie de modul natural de structurare a cunoştinţelor într-un sistem cu inteligenţă umană.

Programarea logică are la bază observaţia unor cercetători că logica simplă sau cel puţin o parte a ei poate fi folosită ca un limbaj de programare. Logica are o interpretare procedurală, care o face adecvată pentru un limbaj în care propoziţiile logice pot fi utilizate ca definiţii sau apeluri de proceduri. Cel mai cunoscut limbaj de acest gen este limbajul PROLOG.

Intr-o aplicaţie reală de modelare pentru un sistem cu comportament complex care presupune luarea unor decizii folosind baze de cunoştinţe, este nevoie să fie utilizate atât metodele modelării dinamice cu ajutorul reţelelor discrete, cât şi metodele orientate pe obiecte cu avantajele transferului ierarhic de informaţii între clase, alături de metodele programării logice, care permite modelarea luării deciziei în condiţii cât mai apropiate de lumea reală. Schema acestui mod de rezolvare a modelării apare în Figura 2.2-1.

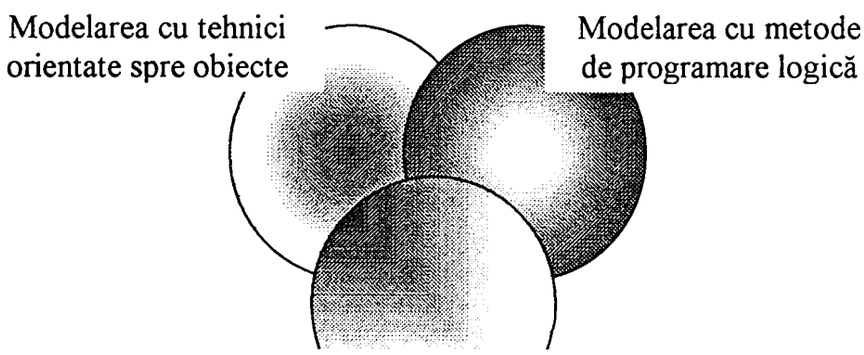


Figura 2.2-1

În zona comună dintre cele trei cercuri s-au dezvoltat o serie de tehnici de programare care conţin elemente de inteligenţă artificială, cum ar fi CAYENE [Ruiz-Mier, S., Talavage, J., 1989] sau PROSS [O'Keefe M., R., 1989].

Problema principală care apare la modelarea cu metoda reţelelor de evenimente discrete sau cu metodele orientate pe obiecte este *problema luării deciziilor în sistem*. O soluţie interesantă în acest sens este prezentată în [Hietiko, E., 1996], unde deciziile în proiectarea unor componente specifice se iau folosind aşa-zisa "*probabilitate subiectivă*",

adică o probabilitate care măsoară încrederea individuală, personală, în evenimentul considerat adevărat. Acest concept este implementat în proiectare cu ajutorul teoremelor lui Bayes. Multe lucrări de referință în domeniul modelării cu elemente de inteligență artificială au folosit de asemenea acest concept [Kaplan, S., Keter, A., Epstain, S., A., ş.a., 1990, Shu, H., 1990]. In capitolul 5 vor fi prezentate principalele elemente teoretice legate de utilizarea teoremelor lui Bayes, precum şi exemple de utilizare pentru domeniul sistemelor flexibile de fabricație. Pe baza acestui concept s-a încercat şi s-au obținut rezultate apreciabile în domeniul *rutării dinamice* într-un sistem de fabricației pe baza unor criterii de decizie "on-line".

După cum s-a văzut anterior, "*simularea*" este procesul de transfer al modelului în lumea reală, adică reproducerea condițiilor de transformări de stare ce apar în realitate. Paşii principali în *desfăşurarea simulării dinamice* [Rumbaugh, J., Blaha, M., ş. a., 1991, pag. 215 şi urm.] sunt:

- *identificarea actorilor simulării*, cum ar fi procesele sau obiectele care sunt luate în considerare şi cărora li se modifică în timp proprietățile;
- *identificarea evenimentelor discrete* şi a succesiunii lor în timp;
- *identificarea dependențelor continui de timp* a evenimentelor şi proceselor şi actualizarea periodică a valorilor parametrilor;
- *definirea buclei de timp* a simulării.

Ca *suport al simulării* se foloseşte de obicei un program de calcul implementat cu ajutorul unui limbaj. Limbajele utilizate frecvent sunt cele din grupa ML (Meta Languages) pentru folosirea modelelor cu rețele discrete, limbajele C++ sau VisualC++ pentru modelarea orientată pe obiecte şi limbajele inteligenței artificiale (limbaje logice de tip PROLOG) pentru simularea logică. Indiferent ce limbaj se va folosi, programul de calcul trebuie să conțină un *editor* care să permită crearea modelului. Funcțiile principale ale editorului, aşa cum rezultă ele din [Rumbaugh, J., Blaha, M., ş. a., 1991, pag. 6 şi urm.] sunt următoarele:

- crearea *modelului static* cu obiecte;
- crearea *modelului dinamic* prin definirea comportamentului în timp al modelului static;
- crearea *modelului funcțional* prin reprezentarea transformărilor valorilor parametrilor din sistem, sub forma unor diagrame de flux de date care conțin noduri (procese) şi arce (fluxuri de date).

Din multitudinea de metode de modelare şi de simulatoare existente în practica de proiectare curentă, un grup important îl constituie cele bazate pe metoda grafurilor. Foarte

multe aplicaţii inginereşti au fost rezolvate cu ajutorul acestor metode, mai ales după apariţia în 1962 a reţelelor Petri. Printre acestea se numără studiul protocoalelor de comunicare în reţelele de calculatoare, sisteme de calcul cu mai multe microprocesoare, studiul reţelelor neuronale, linii şi sisteme de producţie, dar mai ales sisteme flexibile de fabricaţie. De ce sunt reţelele Petri atât de larg utilizate la analiza şi modelarea sistemelor de fabricaţie flexibilă? Iată câteva dintre argumente, aşa cum rezultă ele din [Hrúz, B., 1995] :

- posibilitatea efectivă de a modela interdependenţa dintre evenimente;
- posibilitatea de a modela concurenţa şi sincronizarea evenimentelor;
- gradul înalt de generalitate al metodei.

2.3. Modelarea şi simularea ca etape ale proiectării sistemelor de fabricaţie

Obiectul acestui paragraf îl constituie analiza procesului de proiectare a unui sistem de fabricaţie în scopul identificării locului ocupat de activităţile de modelare şi de simulare în cadrul acestuia. Pe baza teoriei sistemelor prezentată parţial în capitolul anterior, sistemele de fabricaţie fac parte din categoria *sistemelor dinamice cu evenimente discrete* (SDED). Iată în continuare câteva din motivele care stau la baza afirmaţiei de mai sus. Mai întâi se constată că *un sistem flexibil are în componenţă echipamente diferite*, de genul centrelor de prelucrare sau a maşinilor unelte cu comandă numerică, roboţi industriali, sisteme de transport şi de depozitare a materialelor, toate fiind dispuse aleator în spaţiul de lucru pentru diverse layout-uri. In plus, *fiecare componentă a sistemului îndeplineşte în mod individual un anumit program de lucru* localizat la acel nivel. Toate aceste programe, în fapt succesiuni de evenimente discrete, *se înlănţuiesc într-un sistem coerent*, pe baza unei logici implementate la nivelul programului general de conducere al sistemului.

In continuare sunt prezentate câteva dintre evenimentele care se produc într-un sistem flexibil, toate fiind marcate cu un început şi un sfârşit, deci cu un comportament discret in timp:

- intrarea în sistem a unui nou semifabricat;
- transferul unui semifabricat, piesă sau produs de la un post de lucru la altul;
- prelucrarea unei piese sau produs;
- testarea/măsurarea unei piese sau produs;
- ieşirea din sistem a unui produs;
- eliminarea din sistem a deşeurilor, a reziduurilor sau a rebuturilor.

Începutul sau sfârşitul unei operaţii este un *eveniment* în urma căruia starea sistemului se schimbă. *Starea momentană* a sistemului este constituită de desfăşurarea unei operaţii (sistemul este în starea de transfer a unei piese de la unitatea 1 la 2, sau sistemul este în starea de prelucrare a unei piese pe maşina 3, etc.).

Pentru a putea preciza cu mai mare exactitate rolul pe care îl are modelarea şi simularea în proiectarea unui sistem flexibil de fabricaţie este necesar să se facă unele precizări în legătură cu diversele moduri de organizare a producţiei, aşa cum pot fi ele întâlnite în peisajul industrial al zilelor noastre. Un răspuns la întrebarea *"de ce este necesară organizarea producţiei sub formă de sistem flexibil?"* poate fi extras din analiza diagramei prezentată în continuare. Aceasta este răspândită pe scară largă în literatura de specialitate şi redă domeniile de aplicabilitate ale diverselor moduri de organizare a producţiei de bunuri materiale în funcţie de varietatea de repere şi de seria de fabricaţie.

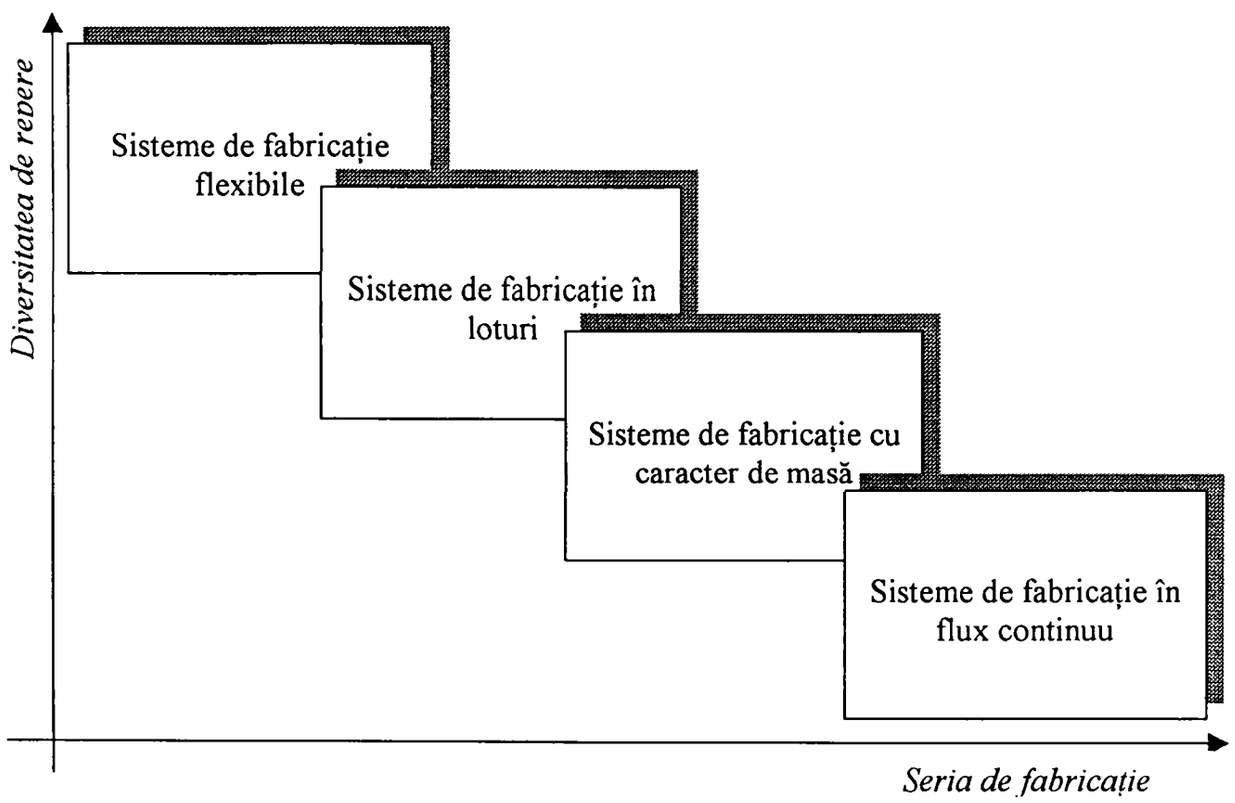


Figura 2.3-1

Este evident faptul că atunci când este vorba de serii mici de fabricaţie, cu o diversitate mare de repere, cel mai indicat mod de organizare al producţiei îl constituie organizarea sub formă de *sisteme flexibile*. Acest mod de organizare fundamental nou, atât ca şi concepţie cât şi ca echipamente şi utilaje folosite, vine în întâmpinarea cerinţelor impuse de piaţă, care stabileşte un ritm deosebit de intens în relaţia dintre producător şi cumpărător, iar "câştigătorul" acestei curse este acel producător care poate oferi cu maximă promptitudine, cu costuri minime şi la

parametrii calitativi ridicaţi ceea ce piaţa pretinde uneori chiar cu rang de unicat! La polul opus se situează organizarea producţiei sub formă de *fluxuri de fabricaţie*, caz în care nu poate fi vorba de nici un fel de flexibilitate, varietatea de repere este practic nulă, iar seria de fabricaţie este infinită. In acest caz se prelucrează de fapt un singur tip de reper, iar trecerea la un alt tip ar presupune mari cheltuieli de pregătire.

In faza de proiectare a unui sistem flexibil de fabricaţie, un rol important revine *modelării şi simulării* ca instrumente de validare prealabilă a unui nou sistem, înainte de efectuarea unor investiţii. Locul modelării şi al simulării în proiectarea unui sistem nou este redat în diagrama din Figura 2.3-2.

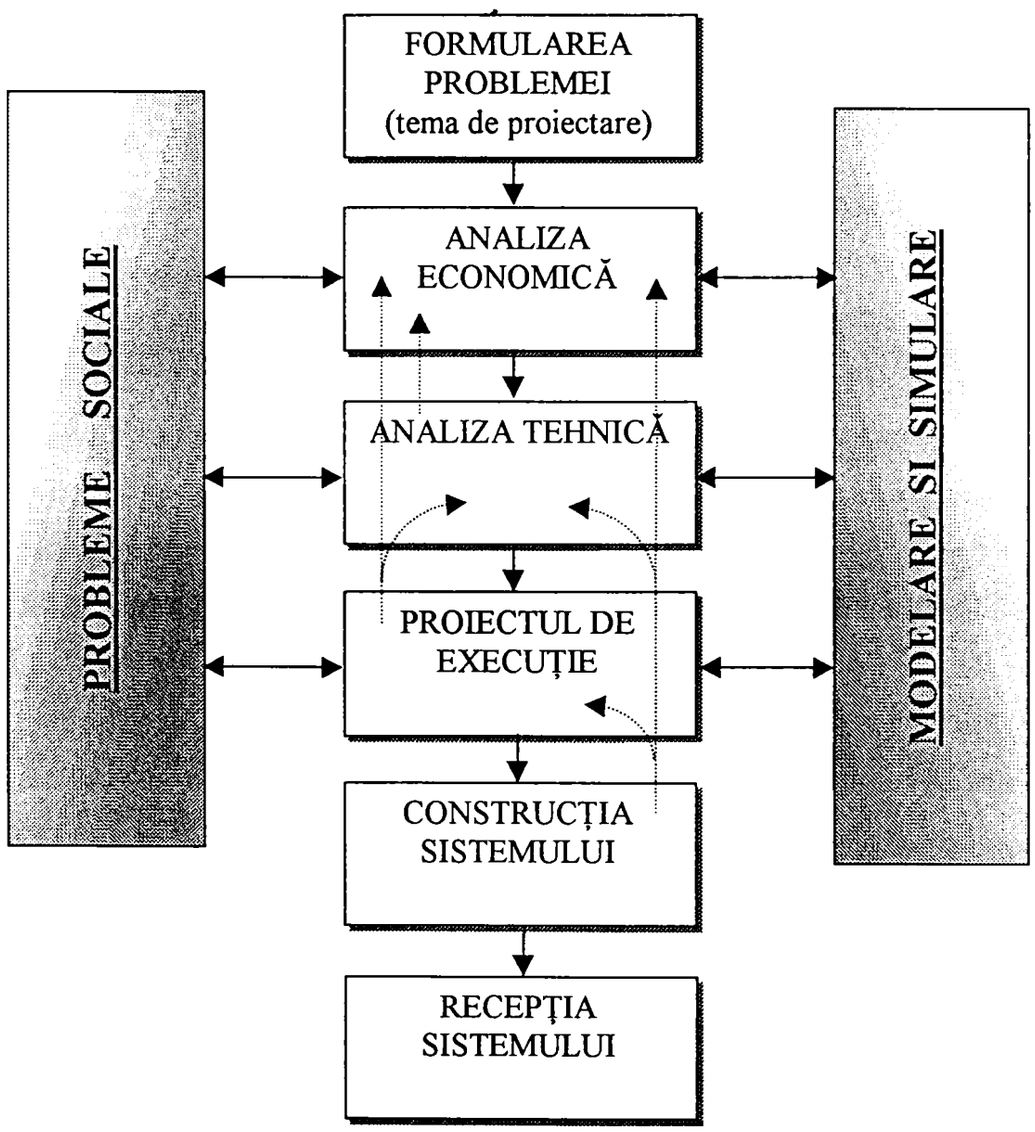


Figura 2.3-2

Analizând diagrama prezentată se pot evidenţia o serie de aspecte relativ la interdependenţa diverselor etape de realizare a unui sistem de fabricaţie. Primul pas în ierarhia deciziei îl constituie *analiza economică* a noului sistem. Aceasta poate releva o eventuală incompatibilitate între necesităţile tehnologice şi de modernizare ale producţiei şi resursele

materiale ale întreprinderii. De asemenea, în această etapă se impune precizarea limitelor financiare între care trebuie să se încadreze noua investiție.

Odată depăşită această etapă se poate trece la *analiza tehnică* a sistemului ce se intenționează a fi realizat. Elementele de studiat în această fază sunt resursele tehnice existente în întreprindere şi care pot fi înglobate în noul sistem, schema de amplasament a utilajelor (layout), schemele de transport, ciclogramele de funcționare pentru diverse tipuri de repere ce se fabrică în sistem, lista utilajelor noi ce vor trebui achiziționate. Toate aceste elemente vor fi în mod obligatoriu corelate cu aspectele financiare.

Etapa următoare se referă la *elaborarea proiectului de execuție* al noului sistem, pe baza analizelor de preț şi a soluțiilor preconizate în etapele anterioare. Soluția finală care va sta la baza construcției noului sistem va determina şi unele reajustări ale soluțiilor tehnice şi financiare din etapele anterioare, închizând astfel o buclă de reacție.

Aspectele sociale ale elaborării unei noi organizări de producție sunt din ce în ce mai importante în condițiile actuale. Atitudinea oamenilor implicați în noua schemă de organizare, atât a celor disponibilizați cât şi a celor reintegrați, poate fi decisiva pentru succesul noului proiect. Noile sisteme de organizare ale producției au în centrul lor omul, ambianța locului de muncă şi calitatea mediului ambiant. Atitudinea conştient-pozitivă a operatorilor umani integrați într-un sistem de fabricație flexibilă se obține de obicei prin tratarea cu maxim interes a acestor aspecte de ordin social.

La acest nivel de elaborare al proiectului unui sistem de fabricație flexibilă apare necesitatea de a utiliza *tehnicile de modelare şi simulare*, care pot valida sau invalida diversele soluții posibile pe baza criteriilor pur tehnice sau a celor tehnico-economice. Marele avantaj al tehnicilor de modelare şi simulare este tocmai posibilitatea pe care o oferă de a testa noul sistem înainte de a începe execuția lui efectivă, deci înainte de a cheltui orice resurse materiale sau financiare. *Modelarea şi simularea ca un proces unic de investigație*, are conexiuni atât cu aspectul financiar al problemei cât şi cu aspectele tehnice şi constructive. In general tehnicile moderne de modelare permit evidențierea unor concluzii globale legate de funcționarea sistemului, concluzii care au un caracter complex, tehnico-economic. Sub forma unor diagrame sau histograme sugestive pot fi apreciate toate elementele de performanță ale sistemului şi pot fi dezvoltate diverse studii comparative între soluțiile posibile din punct de vedere tehnico-economic, cu scopul de a elabora o soluție optimă pe baza unui sistem de criterii.

O tendință actuală în abordarea formelor de organizare a producției de bunuri materiale şi în general a fabricației de produse diverse, o constituie complexul de noțiuni şi

idei organizate în jurul conceptului de *"inginerie concurentă"*. Diversificarea procesului de producție, integrarea lui tot mai accentuată în sistemul general de antreprenoriat, alături de aspectele economice, financiare, de marketing, de prognoză, fiabilitate şi mentenanță, service şi comportament în timp al produselor, este ceea ce marchează în aceste momente evoluția sistemelor industriale. Din ce în ce mai mulți autori se preocupă de aceste noi aspecte ale integrării procesului de fabricație în modul general de dezvoltare al unei întreprinderi [vezi Hietiko, E., 1996].

Locul pe care îl ocupă sistemul de fabricație în conceptul general de "inginerie concurentă" este prezentat în diagrama din figura următoare [Viswanadham, N., Narahari, Y., 1992, figura 2.4].

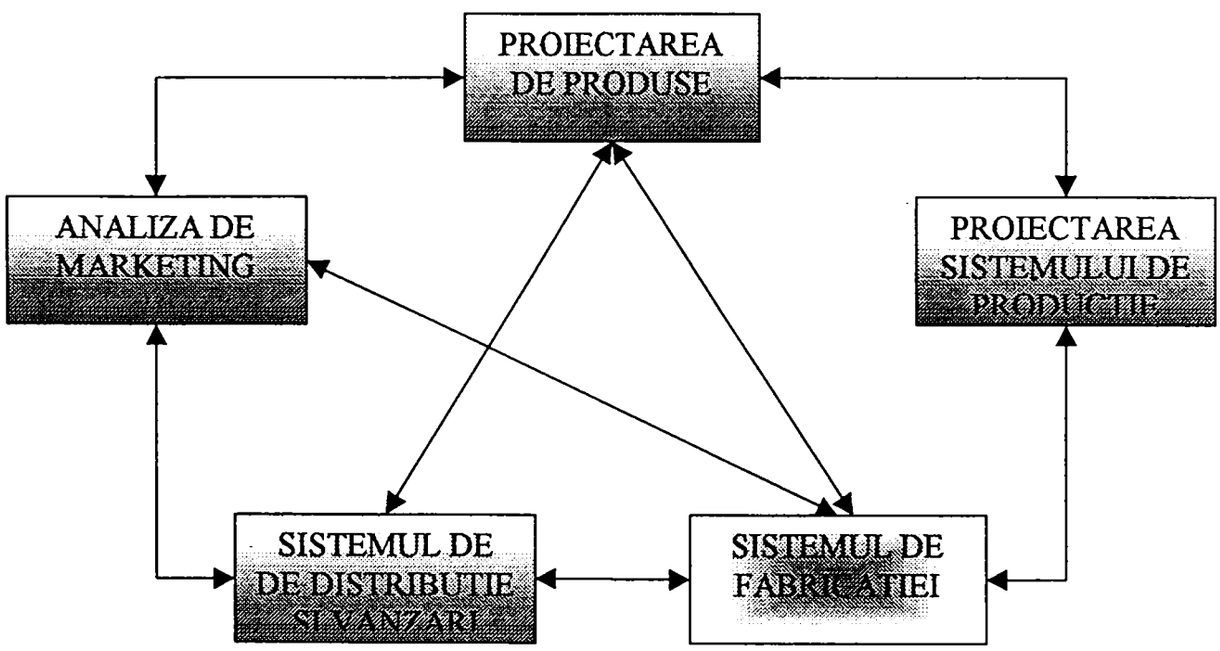


Figura 2.3-3

Analizând această diagramă, se pot observa legăturile bilaterale pe care le are sistemul de fabricație cu toate componentele sistemului antreprenorial. In condiții de concurență acerbă, când nevoile pieței trebuiesc satisfăcute prompt şi în cel mai înalt grad, sistemul de fabricație trebuie să poată să se adapteze în timp real. Fluxurile de informație care se constituie între elementele sistemului asigură coeziunea şi coerența acestuia. Problema deciziei într-un astfel de sistem devine şi mai complexă, modelarea şi simularea având astfel un rol mult mai important.

2.4. Modele de analiză a performanțelor tehnice ale sistemelor de fabricație

Optimizarea funcționării unui sistem de fabricație poate fi realizată prin minimizarea următorilor parametrii:

- timpul total de îndeplinire a tuturor sarcinilor de lucru;
- cantitatea de şpan produs într-un interval de timp în condiții de îndeplinire a sarcinilor de lucru;
- timpul de trecere a unui reper prin sistem;
- numărul de sarcini (task) ce sunt active în sistem la un moment dat;
- întârzierea medie a execuției sarcinilor de lucru.

Aceste criterii de minim se pot constitui în grupuri pentru aprecierea mai fidelă a comportării sistemului, diverse procedee euristice folosind la determinarea unor soluții optime.

Performanțele unui sistem în exploatare pot fi apreciate şi prin alte criterii referitoare la *eficiență şi productivitate*:

- rata de fabricație (repere/oră);
- coeficientul de utilizare al maşinilor;
- probabilitatea ca o maşină să fie blocată cu piese sau complet neîncărcată;
- numărul prezumtiv de piese dintr-un buffer.

2.4.a Lanțuri Markov

Un model Markov este format dintr-un sistem exhaustiv de stări discrete, care descrie toate stările posibile ale sistemului. Intre două stări "i" şi "j" poate avea loc o tranziție cu probabilitatea "p_{ij}".

Starea unui sistem de fabricație format din două maşini şi un buffer poate fi analizată pe baza schemei din figura următoare.

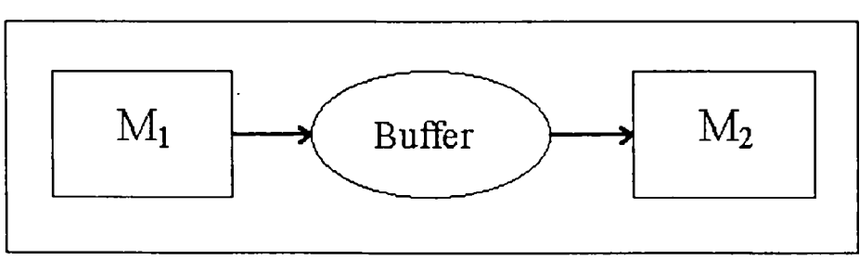


Figura 2.4-1

Parametrii prin care pot fi caracterizate *performanțele unei maşini* integrate într-un sistem de fabricație pe durata unui schimb de lucru sunt:

- timpul de lucru efectiv;
- timpul de reglaj;
- timpul de reparații.

Aceste valori pot fi deterministe sau aleatoare. Schimbările de stare ce apar în interiorul sistemului pot fi declanşate instantaneu, la un anumit moment, caz în care se vorbeşte de *lanțuri Markov cu stări discrete şi cu tranziții discrete*, sau se pot dezvolta în intervale de timp în mod continuu, caz în care se vorbeşte despre lanțuri Markov provenind din *procese continui.*

Pentru a modela starea sistemului din Figura 2.4-1 se va folosi parametrul de stare "s" de valoare:

$$s = (n, \alpha_1, \alpha_2)$$

(2.4-1)

In această relație, "α_1 " şi "α_2 " sunt stările posibile ale celor două maşini, iar "n" este numărul de elemente din buffer adunat cu numărul de piese din maşina 2, ceea ce reprezintă numărul de piese "în ciclu" la un moment dat.

Stările celor două maşini dau următoarele valori parametrului "α"

$$\alpha_i = \begin{cases} 1 \text{ daca masina este activa} \\ 0 \text{ daca masina este in reparatie} \end{cases}$$

(2.4-2)

2.4.a.a. Lanțuri Markov pentru transformări discrete

Sistemul se poate afla la un moment dat în una din stările s_1, s_2, s_3, s_4, ..., s_m. Acest set de parametrii de stare este exhaustiv şi exclusiv mutual, în sensul că el conține toate stările posibile ale sistemului, fără să existe două elemente "s_i" identice. Pentru a aprecia şansele sistemului de a ajunge dintr-o stare în alta se foloseşte noțiunea de *"probabilitate a tranziției"*. Sistemul poate trece din starea "i" de la timpul "k-1" în starea "j" de la timpul "k" în concordanță cu un set de probabilități "p_{ij}".

$$p_{ij} = P\left[s_j(k) \mid s_i(k-1)\right] \quad i \le i, j \le m$$

(2.4-3)

Condiția ca lanțul Markov să fie cu stări discrete se poate formula în felul următor:

$$P\left[s_j(k) \mid s_a(k-1), s_b(k-2), s_c(k-3)\ldots\right] = P\left[s_j(k) \mid s_a(k-1)\right]$$

(2.4-4)

Din analiza acestei relații se poate deduce că probabilitatea de a ajunge din starea "a" în starea "j" trecând prin stările "b", "c", etc., este egală cu probabilitatea de trecere directă din starea "a" în starea "j", cu alte cuvinte sistemul nu are memorie şi toate stările de dinaintea stării "a" nu au nici o relevanță pentru trecerea sistemului în starea "j".

De cele mai multe ori, în practică, probabilitatea ca sistemul să treacă dintr-o stare în alta parcurgând un număr "n" de paşi nu depinde de momentul "t" de plecare. Acest lucru este exprimat de relația

$$p_{ij}(t, t+n) = p_{ij}(n)$$

(2.4-5)

Probabilitățile "p_{ij}" pentru un sistem cu "m" stări posibile se pot ordona într-o *matrice a probabilităților tranzițiilor unitare de stare* "P", unde suma probabilităților pe fiecare linie va fi 1, iar probabilitățile vor fi independente de timpul "k" la care apare tranziția:

$$P = \begin{pmatrix} p_{11} & p_{12} & p_{1m} \\ p_{21} & p_{22} & p_{2m} \\ & & \\ p_{m1} & p_{m2} & p_{mm} \end{pmatrix}$$

(2.4-6)

Pe baza ecuației Chapman – Kologorov [vezi Viswanadham, N., Narahari, Y., 1992, pag. 188 şi urm.] se poate calcula probabilitatea tranzițiilor compuse:

$$p_{ij}(m+n) = \sum_{k \in S} p_{ik}(m) p_{kj}(n)$$

(2.4-7)

Pornind de la această relație se poate deduce faptul că matricea de probabilitate după "k" tranziții este egală cu puterea "k" a matricii de probabilitate a tranzițiilor unitare. Ecuația următoare exprimă această concluzie:

$$P(k) = P(0)P^k \quad k = 0,1,2,\ldots$$

(2.4-8)

In această relație P(0) este matricea probabilităților tranzițiilor unitare la momentul "0".

2.4.a.b. Lanțuri Markov pentru transformări continui

De cele mai multe ori tranzițiile dintr-un sistem de fabricație, erorile de prelucrare sau defecțiunile utilajelor intervin la intervale neregulate de timp, distribuite aleator, ceea ce face

necesară luarea în considerare a comportamentului continuu al sistemelor, în locul comportamentului discret. Lanțurile Markov de timp continuu folosesc noțiunea de *coeficient de tranziție*, în locul noțiunii de probabilitate a tranziției. Prin definiție, *coeficientul de tranziție "a_{ij}"* reprezintă rata sau proporția tranzițiilor din starea "i" în starea "j". Pentru procese stabile, coeficientul de apariție al unei stări este egal cu coeficientul de dispariție al acesteia, (coeficientul de reversie) ($a_{ij} = a_{ji}$) pe această bază putându-se scrie ecuațiile globale de echilibru ale sistemului. Coeficientul de tranziție se exprimă printr-un produs dintre probabilitatea de apariție a stărilor stabile "p_i" și un coeficient de pondere λ_i.

Noțiunile de mai sus vor fi exemplificate pe un sistem model, cu trei stări, conform schemei din Figura 2.4-2.

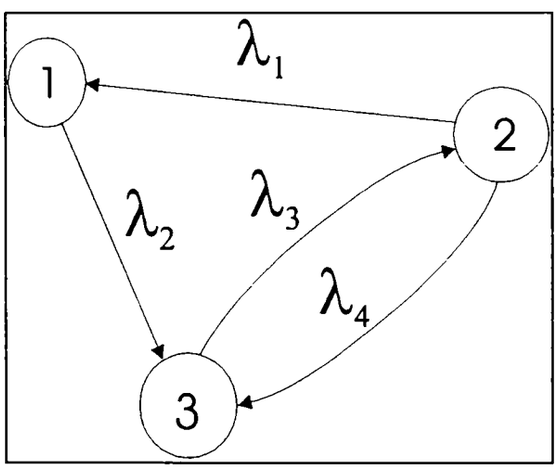


Figura 2.4-2

Ecuațiile globale de echilibru ale acestui sistem în ipoteza stabilității vor fi următoarele:

$$\begin{cases} \lambda_2 p_1 = \lambda_1 p_2 \\ \lambda_1 p_2 + \lambda_4 p_2 = \lambda_3 p_3 \\ \lambda_3 p_3 = \lambda_4 p_2 + \lambda_2 p_1 \end{cases}$$

(2.4-9)

Este evident că suma probabilităților de stare va trebui să fie egală cu unitatea, deci:

$$p_1 + p_2 + p_3 = 1$$

(2.4-10)

Dacă se notează cu "π" matricea probabilităților, sistemul de mai sus se poate scrie sub forma:

$$\pi \begin{bmatrix} -\lambda_2 & 0 & \lambda_2 \\ \lambda_1 & -(\lambda_1 + \lambda_4) & \lambda_4 \\ 0 & \lambda_3 & -\lambda_3 \end{bmatrix} = 0$$

(2.4-11)

Rezolvând acest sistem în raport cu "π" se pot deduce o serie de informații cu privire la performanțele sistemului, cum ar fi productivitatea acestuia, gradul de utilizare al maşinilor, numărul probabil de piese din buffer, etc.

2.4.b Rețele de şiruri de aşteptare

Pentru a identifica şi detalia principalele elemente ale unei rețele de şiruri de aşteptare, să considerăm următorul exemplu: rețeaua este formată din "N" noduri sau posturi de lucru (maşini unelte, buffere, transportoare, puncte de măsurare, etc.), starea rețelei la un moment dat este descrisă de vectorul $k=(k_1, k_2, k_3, \ldots k_N)$, unde "$k_i$" este numărul de clienți (piese) la postul respectiv.

Rata de sosire a pieselor în sistem "λ" depinde de numărul total de piese din sistem şi este 0 dacă acest număr depăşeşte o anumită valoare "k_0" considerată numărul maxim de piese pe care îl poate accepta sistemul. Se poate afirma deci că numărul total de piese din sistem, "K", va fi:

$$K = \sum_{i=1}^{N} k_i \qquad \textbf{(2.4-12)}$$

Rata de ieşire a pieselor din sistem "μ" , depinde de regulă de indicele "n" al postului din rețea şi de mărimea şirului de aşteptare aferent acestuia, sub forma $\mu = \mu(n, k_n)$, n= 1,...,N.

Deplasarea pieselor prin rețea este modelată sub forma unui parcurs aleator între diversele posturi de lucru, trecerea unei piese de la postul "i" la "j" fiind caracterizată de probabilitatea "p_{ij}" (vezi Figura 2.4-3). Matricea R= r(i,j), unde i=0,...,N şi j=1,...,N+1 este *matricea de rutare* a sistemului.

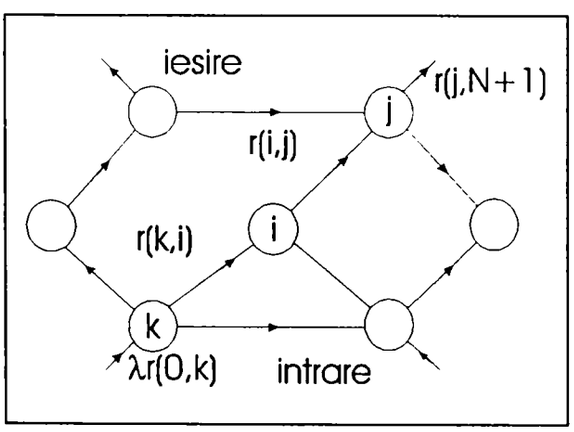


Figura 2.4-3

Pornind de la *regula de conservare a fluxului de piese*, care afirmă că numărul de piese care intră într-un nod este egal cu numărul de piese care iese din acel nod (sub aspect material) după prelucrare, se poate determina numărul de piese ce tranzitează nodul "l" cu relația:

$$T_l = \lambda r(0,l) + \sum_{j=1}^{N} r(j,l) T_j$$

(2.4-13)

In această relație "r(0,l)" este probabilitatea ca postul "l" să fie primul din rețea, iar "T_j" este tranzitul de piese prin postul "j". După cum se afirmă în [Hauke Jungnitz, Alan Desrochers, 1991, pag. 74] o soluție stabilă a acestei probleme este de forma:

$$\pi(\bar{k}) = \prod_{i=1}^{N} f_i(k_i)$$

(2.4-14)

unde $f_i(k_i)$ este probabilitatea de a găsi "k_i" repere într-un şir de aşteptare la postul "i".

Un domeniu de perspectivă aparținând rețelelor de şiruri de aşteptare îl constituie situația în care reperul sau piesa prelucrate în sistem îşi schimbă calitatea pe durata tranzitului, obținându-se astfel mai multe repere, de tip diferit, cu trasee distincte în interiorul rețelei. Probleme de acest gen au fost abordate de Baskett, Muntz şi Palacios.

2.4.c Rețele Petri stocastice generalizate

Rețeaua Petri este un graf format din două tipuri de elemente: *poziții şi tranziții*. Pozițiile sunt elementele rețelei care reprezintă (modelează) condițiile necesare pentru declanşarea unui anumit eveniment, iar tranzițiile modelează evenimentele care se produc la îndeplinirea anumitor condiții reprezentate de pozițiile de pe arcele de intrare. Relațiile dintre poziții şi tranziții sunt reprezentate prin arcele orientate ale grafului [Francisc Kovacs, Sanda Grigorescu, Cornel Rădulescu, 1994, pag. 148 şi urm.]

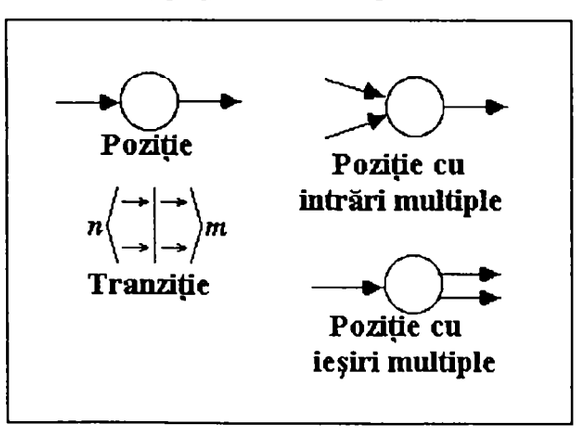


Figura 2.4-4

In Figura 2.4-4 sunt reprezentate pozițiile şi tranzițiile din rețea aşa cum apar ele schematic în graful de descriere a rețelei. [în Douglas Lyon, 1996, pag. 3 şi urm.]

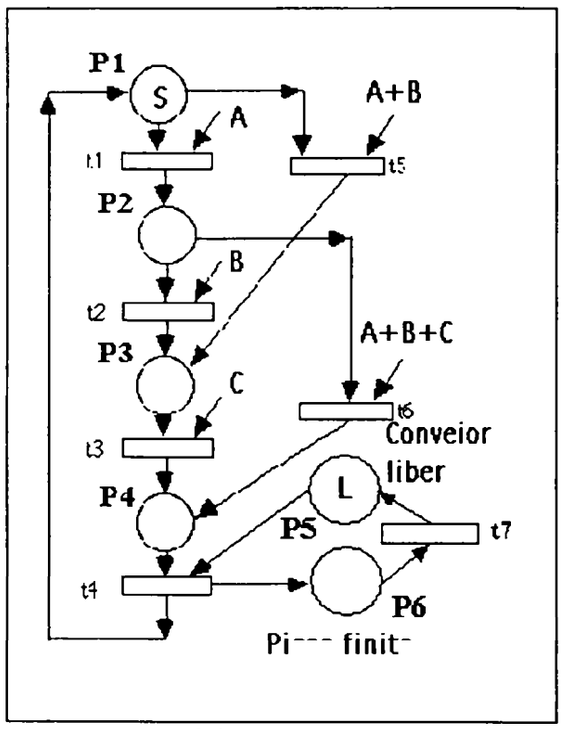


Figura 2.4-5

Relațiile care se stabilesc între poziții şi tranziții pot fi reprezentate şi printr-un tabel sintetic, ale cărui avantaje se remarcă în procesul de calcul. Fie de pildă rețeaua Petri din figura alăturată, care modelează funcționarea unui automat de asamblare. Acesta realizează montarea pe un şasiu "S" a unui sistem de trei piese notata cu A, B şi C. Piesele pot fi adăugate pe rând sau în grupuri de câte două, astfel încât în final să se obțină sistemul "S+A+B+C". Pentru a putea trece la pasul următor, echivalent cu adăugarea unei noi piese la ansamblul existent la un moment dat, trebuie mai întâi ca procesul să fie inițializat. La nivelul fiecărei poziții se cercetează componența ansamblului existent, după care se produce tranziția spre poziția următoare, unde se verifică din nou componența ansamblului şi tot aşa, până când se obține totalul de piese care va declanşa tranziția de evacuare a ansamblului din sistem cu condiția să existe un conveior liber. Aceste activități sunt sintetizate în Tabelul 2.4-1.

Intr-un mod mai general [în Hauke Jungnitz, Alan Desrochers, 1991, pag. 76 şi urm.] se defineşte o *rețea Petri ca un graf* reprezentat printr-un cvadruplu N=(P, T, I, O), unde:

- $P=\{p_1, p_2, \ldots, p_r\}$ este un set de poziții;
- $T=\{t_1, t_2, \ldots, t_q\}$ este un set de tranziții (sau bare);
- I: $P \times T \rightarrow \aleph$ ($\aleph$ este mulțimea numerelor naturale) este o funcție de intrare (input) care defineşte mulțimea arcelor orientate de la P la T;
- O: $T \times P \rightarrow \aleph$ este o funcție de ieşire (output), care defineşte mulțimea arcelor orientate de la T la P.

Mulțimile P şi T sunt disjuncte, deci $P \cap T = \varnothing$. Arcele orientate pot avea gradul de multiplicitate mai mare ca unitatea, în acest caz se spune că rețeaua Petri este cu arce multiple.

Tabelul 2.4-1

Conținut	**Poziția**	**Decizie**	**Tranziție**	**Acțiune**	**Poziția următoare**
S	P1	A	t1	S+A	P2
S	P1	A+B	t5	S+A+B	P3
S+A	P2	B	t2	S+A+B	P3
S+A	P2	B+C	t6	S+A+B+C	P4
S+A+B	P3	C	t3	S+A+B+C	P4
S+A+B+C	P4	Conv. liber	t4	Evacuare ans.	P6, P1
Ansamblu	P6	Evacuare	t7	Eliberare conv.	P5
S	P1	A	t1	S+A	P2

Îndeplinirea condițiilor de realizare a unui eveniment (tranziție) este simbolizată prin prezența unui jeton (marcaj) peste poziția de origine sau de intrare. Dacă rețeaua este cu arce multiple şi marcajele pot fi multiple, ceea ce conduce la *definiția generală a marcajului* ca o funcție definită pe mulțimea pozițiilor P, cu valori în mulțimea numerelor naturale, $M : P \rightarrow \aleph$.

Pentru o anumită stare a sistemului, marcajul poate fi definit ca un vector ce conține toate seturile de jetoane din cele "r" poziții din graf $M=(M(p_1), M(p_2),\ldots, M(p_r))$. După executarea unei tranziții marcajul dintr-o poziție dată "p_i" se va schimba după regula:

$$M_1(p_i) = M_0(p_i) + O(t_j, p_i) - I(p_i, t_j)$$

(2.4-15)

Se pot defini două matrici de dimensiune "r×q" care să descrie funcțiile de intrare, respectiv cele de ieşire. Matricea C^- (matrice de intrare sau *matrice PRE*) conține în poziția "(i,j)" ordinul de multiplicitate al arcului ce leagă tranziția "j" de poziția de intrare "i". Matricea C^+ (matrice de ieşire sau *matrice POST*) conține în poziția (i,j) ordinul de multiplicitate al arcului ce leagă tranziția "j" de poziția ei de ieşire "i". Aceste matrici se numesc *matrici de marcaj*. Dacă marcajele dintr-o poziție oarecare a grafului au doar valorile 0 sau 1 (fiecare poziție conține cel mult un jeton), rețeaua se numeşte *rețea binară*. Cu ajutorul matricilor PRE şi POST se poate calcula *matricea de incidență* "C" a unei tranziții:

$$C = C^+ - C^-$$

(2.4-16)

Pentru ca o rețea Petri să modeleze corect un proces de fabricație real, ea va trebui să îndeplinească unele *condiții de validare*.

O primă condiție este ca rețeaua *să fie viabilă*, adică să conțină numai tranziții viabile. Tranziția viabilă se definește ca fiind acea tranziție care se regăsește în orice secvență "S" de tranziții ce pornește de la o matrice de marcaj "M_i" dată. Altfel spus, rețeaua este viabilă dacă pentru orice matrice de marcaj dedusă din cea de start după executarea unui număr de tranziții, există o succesiune de tranziții executabile, deci rețeaua nu se blochează.

O altă condiție necesară pentru existența și funcționarea unei rețele Petri este ca rețeaua *să fie proprie (reinițializabilă)* , ceea ce înseamnă că din orice matrice de marcaj existentă la un moment dat să existe o secvență de tranziții care să conducă la matricea de marcaj inițială.

Ideea modelării evoluției sistemelor cu ajutorul rețelelor Petri este de a urmări mișcarea jetoanelor din poziții, deci de a urmări cum evoluează acestea după trecerea unei secvențe de tranziții. Dacă se definește un vector "y" care să conțină numărul de incidențe ale tranzițiilor dintr-o secvență dată, atunci matricea de marcaj finală "M_n" se poate obține din cea inițială "M_0" cu ajutorul relației:

$$M_n = M_0 + C \cdot y$$

(2.4-17)

Problematica rețelelor Petri prezentată până la acest punct se bazează pe prezumția că tranzițiile se declanșează instantaneu atunci când sunt îndeplinite condițiile necesare, fără nici un fel de întârziere, după modelul sistemelor electronice. In realitate însă, mai ales când este vorba de sisteme mecanice, cu un timp de răspuns mai îndelungat, cum ar fi durata transportului unei piese dintr-un buffer la o mașină unealtă, sau durata prelucrării unui reper la un centru de prelucrare, timpul de realizare a unei tranziții nu mai poate fi ignorat. Se ajunge astfel la *rețelele Petri temporale*, la care tranzițiile se realizează în intervale de timp cu distribuție exponențială. Acest gen de distribuție de probabilitate este singurul care realizează condiția de independență față de trecut, adică probabilitatea de realizare a unei tranziții nu depinde de parcursul pe care l-a avut sistemul până în acel punct (vezi § 2.4.a). Funcția care modelează densitatea de probabilitate este de forma:

$$f(x) = \mu \exp(-\mu x) u(x)$$

(2.4-18)

unde u(x) este funcția treaptă.

In practică se întâlnesc de obicei situații complexe, când unele tranziții au caracter atemporal, deci se produc instantaneu, iar altele au caracter temporal. Pe graful rețelei arcele temporale se vor desena cu linie întreruptă. In abordarea acestor situații se pune problema de a găsi densitatea de probabilitate a producerii fiecărei tranziții, pornind de la un set de

parametrii de calcul cu caracter de coeficienţi de pondere $W=(\omega_1, \omega_2, \omega_3, \ldots, \omega_r)$ referitor la cele "r" tranziţii posibile, precum şi de la un set de coeficienţi de prioritate ai tranziţiilor, $R=(r_1, r_2, r_3, \ldots, r_r)$. Pe baza acestor elemente, cu ajutorul unor programe specializate, se pot deduce parametrii de performanţă ai sistemului, cum ar fi rata de transfer, coeficienţii de utilizare ai utilajelor, timpii de aşteptare în buffere, etc.

2.4.d Reţele Petri colorate

Pornind de la structura şi proprietăţile reţelelor Petri, se observă că în unele situaţii numărul de parametrii cu care operează aceştea poate fi insuficient pentru modelarea completă a unui sistem flexibil. Operând doar cu marcajele de la nivelul poziţiilor şi neluând în considerare diversele aspecte particulare ale realizării unei tranziţii, aceste reţele nu pot cuprinde nuanţe de genul schimbărilor de scule, al defecţiunilor şi reparaţiilor de utilaje, etc. [în H. Alla, P. Ladet, 1986, pag. 271 şi urm.]. Din acest motiv s-a impus dezvoltarea unui nou tip de reţele Petri, numite *"reţele Petri colorate"*, care se deosebesc de cele binare prin faptul că o poziţie oarecare p[oate conţine jetoane de tipuri diferite, aceste tipuri fiind denumite "culori". De aici provine şi denumirea acestui nou tip de reţele. Acestea pot realiza mai multe genuri de utilităţi: modelare, verificare şi simulare a unor sisteme de fabricaţie flexibilă, etc.

Definiţia formală a unei reţele Petri colorate este:

$$N = (P, T, C, I, O, M_0)$$

(2.4-19)

unde P, T, I, O, au semnificaţiile din definiţia dată anterior. C(p) şi C(t) sunt *seturi de culori (colors)* atribuite poziţiilor şi tranziţiilor. M_0 este marcajul iniţial. *O tranziţie este iniţializată atunci când toate poziţiile de intrare ale ei conţin cel puţin atâtea jetoane din fiecare culoare câte sunt cerute de inscripţiile de pe arcele ei de intrare*. Astfel se declanşează tranziţia respectivă care va determina modificarea numărului de jetoane ale poziţiilor de ieşire şi ale celor de intrare în conformitate cu inscripţiile de pe arce. In acest fel se va ajunge la un nou marcaj. Analitic, condiţia de validare a unei tranziţii pentru o anumită culoare se poate scrie:

$$M(p)(c) \geq I(p,t)(c) \qquad \forall p \in \aleph$$

(2.4-20)

Un exemplu de *modelare* cu reţele Petri colorate îl constituie cazul unui buffer de piese diferite, care funcţionează pe principiul FIFO (first in - first out). Graful reţelei poate fi observat în Figura 2.4-6, bufferul fiind dimensionat pentru "n" piese cu distribuţia "m_j" $j \in [1,p]$, "p" fiind numărul variantelor de repere diferite.

Setul de culori se poate exprima prin:

$$C = \{\langle m_j, e_k\rangle \cup \langle m_j\rangle; k \in [1,n], j \in [1,p]\}$$

(2.4-21)

Un marcaj "e_k" în poziția "G" din şir înseamnă că poziția respectivă este neocupată ($m_j = 0$). Dacă $m_j \neq 0$ se poate afirma că în poziția respectivă există un marcaj, care poate determina o tranziție.

Se definesc funcțiile "u" şi "v", care atribuie fiecărui tip de produs poziția primă respectiv ultimă din şir, cu relațiile:

$$u : (m_j) \rightarrow \{\langle m_j, e_1\rangle\} / u(m_j) = \langle m_j, e_1\rangle$$
$$v : (m_j) \rightarrow \{\langle m_j, e_n\rangle\} / v(m_j) = \langle m_j, e_n\rangle$$

(2.4-22)

Se vor defini în continuare funcțiile "f" şi "g", care realizează testarea umplerii unui locaş din şir, respectiv avansul cu o poziție în şirul de produse:

$$f : \{\langle m_j, e_k\rangle\} \rightarrow (e_k) \ / \ f(\langle m_j, e_k\rangle) = e_k, k \in [1,n]$$
$$g : \{\langle m_j, e_k\rangle\} \rightarrow \{\langle m_j, e_k\rangle\} \ / \ g(\langle m_j, e_k\rangle) = \langle m_j, e_{k+1}\rangle \ k \in [1,n-1]$$

(2.4-23)

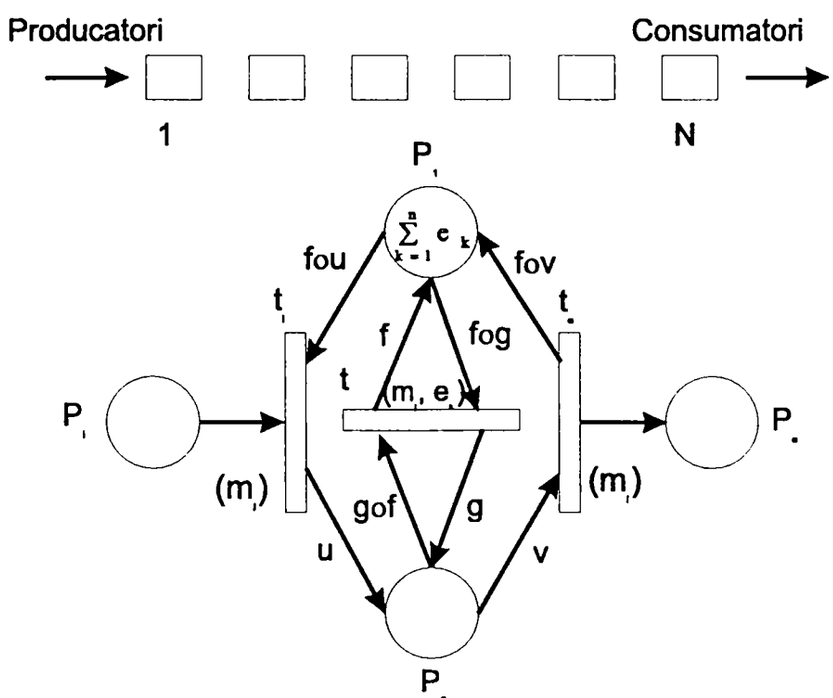


Figura 2.4-6

Setul de culori asociat pozițiilor pline din şir este definit de "Q", iar cel asociat pozițiilor goale, de "E". Astfel, pentru marcajul inițial, cu toate pozițiile goale, se poate scrie:

$$M_o(Q) = 0 \text{ si } M_0(E) = \sum_{k=1}^{n} e_k$$

(2.4-24)

Tranziția de intrare "t_i" este posibilă raportat la culoarea "m_j", dacă poziția "E" conține un marcaj de culoarea $f \circ u(m_j) = e_1$, ceea ce înseamnă că poziția "1" din şir este liberă. Efectuarea tranziției "t_i" determină apariția unui marcaj de culoare $u(m_j)=\langle m_j,e_1\rangle$ în poziția "Q" , elementul "1" conținând un marcaj de tip "j". Acest marcaj va fi transferat de la un element la următorul prin efectuarea tranziției "t". Să presupunem că poziția "Q" conține un marcaj $\langle m_j,e_k\rangle$. Tranziția "t" va fi executată conform culorii $\langle m_j,e_k\rangle$ dacă poziția "E" conține un marcaj de culoare $f \circ g(\langle m_j,e_k\rangle) = e_{k+1}$, ceea ce înseamnă că poziția "k+1" trebuie să fie liberă. Efectuarea tranziției "t" determină apariția unui marcaj de culoare $f(\langle m_j,e_k\rangle)=e_k$ în poziția "E" (poziția "k" este liberă) şi un marcaj de culoare $g(\langle m_j,e_k\rangle)=\langle m_j,e_{k+1}\rangle$ în poziția "Q", deci poziția "k+1" din şir va fi ocupată. Acest proces se va desfăşura atâta timp cât va mai exista loc liber în şir, deci până la umplerea acestuia şi apariția unui marcaj în poziția 'n". Apoi reperul din această poziție va fi evacuat prin tranziția de ieşire "t_o", astfel încât ultima poziție din şir va rămâne liberă.

Ca instrument de *validare* al unui sistem de fabricație, rețelele Petri colorate se bazează pe determinarea invarianților liniari ai marcajelor. Aşa după cum s-a arătat mai devreme, (vezi ecuația (2.4-17)), marcajul existent după efectuarea unei secvențe "s" pornind de la M_0, se poate scrie funcție de marcajul de pornire, de matricea de incidență şi de un vector "y" care conține informații despre numărul de incidențe ale fiecărei tranziții din secvența "s". Dacă se consideră un vector "J" care are proprietatea că $J^t \bullet C=0$, atunci acest vector este *un invariant liniar al matricii marcajelor*. Pe această bază, combinat cu relația $M_n = M_0 + C \cdot y$), se va obține:

$$J^t \cdot M = J^t \cdot M_0 + J^t \cdot C \cdot y = J^t \cdot M_0$$

(2.4-25)

Se observă în expresia de mai sus că membrul din dreapta al egalității este constant, el depinzând doar de marcajul inițial. Existența unor astfel de invarianți este sursa unor concluzii referitoare la proprietățile naturale ale unei rețele, cum ar fi mărginirea, lipsa blocajelor, viabilitatea, etc.

Aplicând teoria de mai sus la exemplul prezentat anterior, matricea de incidență a rețelei se va putea scrie:

$$C = \begin{matrix} Q \\ E \end{matrix} \begin{bmatrix} \overset{t_i}{u} & \overset{t}{g - id} & \overset{t_o}{-v} \\ -f \circ u & f - f \circ g & f \circ v \end{bmatrix}$$

(2.4-26)

Soluția unică a ecuației $J^t \bullet C = 0$ pentru forma de mai sus a matricei de incidență este $J^t = [f \quad id]$, unde față de notațiile cunoscute, "id" este funcția identică. Invariantul obținut astfel conduce la următoarele ecuații:

$$[f \quad id] \cdot \begin{vmatrix} M(Q) \\ M(E) \end{vmatrix} = [f \quad id] \cdot \begin{vmatrix} M_0(Q) \\ M_0(E) \end{vmatrix} = [f \quad id] \cdot \begin{vmatrix} 0 \\ \sum_{k=1}^{n} e_k \end{vmatrix}$$

$$\Rightarrow f(M(Q)) + M(E) = \sum_{k=1}^{n} e_k$$

(2.4-27)

Acest rezultat poate fi interpretat în felul următor: dacă poziția "E" conține un marcaj "e_k", aceasta înseamnă că nu există nici un marcaj de tipul $\langle m_j, e_k \rangle$ în poziția "Q". Elementul "k" din şir este gol, ceea ce concordă cu semnificația poziției "E" (poziție liberă); elementul "k" poate conține un singur marcaj de indice "k". In caz contrar transformata prin funcția "f" ne-ar conduce la rezultatul $2e_k$, ceea ce este în contradicție cu membrul doi al relațiilor de mai sus.

Mai mult, se poate demonstra că:

- rețeaua este mărginită pentru $M(E) < \sum_{k=1}^{n} e_k$ si $M(Q) < \sum_{k=1}^{n} \langle m_j, e_k \rangle$;
- rețeaua nu se poate bloca.

Să presupunem că M este un marcaj posibil pornind de la cel inițial. Din expresia invariantului se poate deduce:

- $M(E) = \sum_{k=1}^{n} e_k$, deci tranziția de intrare t_i este posibilă;
- $M(E) = 0$, deci $M(Q) = \sum_{k=1}^{n} \langle m_j, e_k \rangle$ şi deci tranziția de ieşire t_o este posibilă;
- $M(E) = e_{k1} + e_{k2} + e_{k3} + ... + e_{ks} \quad s \in [1, n-1]$ Din această prezentare se pot trage două concluzii referitoare la blocajele din sistem:
- $\forall r \in [1,s], \quad k_r \neq n$, Q conține $\langle m_j, e_n \rangle$ şi tranziția "t_o" este declanşabilă;
- $\exists r \in [1,s], \quad k_r = n$, atunci $k \in [1,n-1]$ astfel încât Q conține pe $\langle m_j, e_k \rangle$ şi nu conține pe $\langle m_j, e_{k+1} \rangle$ şi tranziția "t" este executabilă.

Din relațiile de mai sus se poate deduce că sistemul nu prezintă blocaje.

Rețelele Petri colorate pot fi utilizate și ca *mijloc de simulare* a funcționării unui sistem. In acest caz se va ține cont și de variabila timp în desfășurarea tranzițiilor. In mod concret timpul apare în calcule prin intermediul unei întârzieri "d" asociată fiecărui marcaj de o anumită culoare specific unei anumite poziții. Această întârziere poate reprezenta un timp de prelucrare, un timp de așteptare într-un buffer sau un timp de transport, sau orice alt interval de timp consumat prin desfășurarea unei transformări de stare a sistemului. Spre deosebire de rețelele Petri obișnuite, *rețelele Petri temporale* nu permit validarea unei tranziții până nu se derulează intervalul de timp "d" specific marcajului în care are loc validarea.

2.5. Prezentarea unor metode și programe de simulare pentru sisteme cu evenimente discrete

Paragraful de față se ocupă de prezentarea succintă a unor programe folosite de autor în faza de documentare și de elaborare a lucrării de doctorat, sau cu ocazia stagiilor de pregătire desfășurate în străinătate.[a] Programele au fost grupate în două mari categorii, după metoda de simulare folosită:

- programe care folosesc pentru simulare *grafica animată*;
- programe de simulare simbolică bazate pe *diagrame active*.

Deosebirea esențială dintre cele două categorii constă în felul în care se face simularea, adică în modul de aducere a modelului în realitate. Dacă programele de animație folosesc medii grafice 2D sau 3D sau chiar realitatea virtuală pentru a "vedea" ce se întâmplă în realitate în sistemul modelat, celelalte programe, bazate pe diagrame (rețele) active, folosesc pentru simulare mișcarea unor jetoane între pozițiile de pe diagramă, de-alungul unor arce și trecând prin tranziții care se validează în anumite condiții, schimbând astfel starea sistemului.

In general, toate programele de simulare prezentate fac parte din categoria simulatoarelor pentru sisteme cu evenimente discrete, acestea fiind cele mai potrivite pentru abordarea sistemelor de fabricație. Totuși, există situații în care pe fondul unei evoluții cu evenimente discrete apar transformări continui, care pot fi descrise de ecuații matematice în care parametrii sistemului evoluează continuu în timp. Unele din programele ce se prezintă în continuare au astfel de posibilități.

[a] August-Octombrie 1993 și Septembrie-Decmbrie 1996, la Kuopio Institute of Technology (KUTOL), Finlanda

2.5.a Modelarea şi simularea cu ajutorul programelor de grafică animată

2.5.a.a. Utilizarea grupului de programe QUEST

Pachetul de programe de modelare – simulare oferit de firma Deneb[b] se numără printre cele mai performante aplicaţii de acest fel existente pe piaţa de software la ora actuală. Programele rulează pe staţii grafice UNIX sau HPIX, precum şi pe platforme WindowsNT. Acest pachet integrat conţine următoarele componente:

- ENVISION® folosit ca mediu virtual pentru proiectare, analiză şi validarea unor soluţii;
- IGRIP - pentru proiectare interactivă de procese şi procese;
- Virtual NC® utilizat la modelarea şi simularea proceselor de prelucrare pe maşini unelte cu comandă numerică (vezi Figura 2.5-1);
- QUEST® folosit pentru simularea sistemelor cu evenimente discrete;

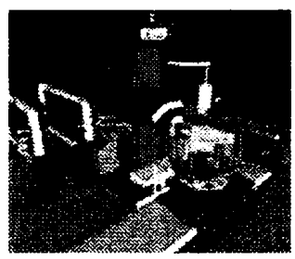


Figura 2.5-1

- Deneb/ERGO destinat simulării spaţiilor de lucru cu operatori umani;
- UltraArc® folosit la simularea şi programarea off – line a roboţilor de sudare cu arc electric (vezi Figura 2.5-2);

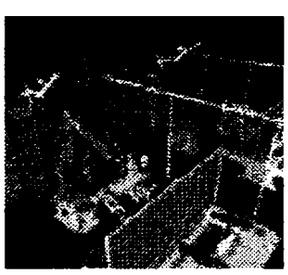


Figura 2.5-2

- UltraGRIP® conceput pentru modelarea şi simularea unei largi palete de aplicaţii robotizate de diverse genuri;
- UltraPaint® destinat aplicaţiilor de vopsire robotizată;
- UltraSpot® utilizat în programarea roboţilor de sudare în puncte.

Programul de modelare şi simulare Quest este destinat sistemelor cu evenimente discrete din care fac parte şi sistemele de fabricaţie. Acest program se bucură de câteva facilităţi de o mare utilitate, printre care se pot enumera:

- o logică naturală asociată implicit obiectelor uzuale din spaţiul de lucru (centre de prelucrare, operatori umani, sisteme de transport, etc.) precum şi unor strategii tipice de conducere integrată a sistemului (metoda "Just in Time", metoda Kanban);
- posibilitatea de a controla proprietăţile fizice ale sistemului, cum ar fi vitezele de deplasare ale mijloacelor de transport, distanţele de parcurs între posturile din sistem, masele şi dimensiunile pieselor, traiectoriile de mişcare ale elementelor mobile;

[b] URL: http://www.deneb.com

- cu ajutorul unui "Batch Control Language" se pot configura diverse scenarii de simulare precum şi diverse modalităţi de colectare şi de utilizare a rezultatelor simulării;
- prin folosirea limbajului structurat "Simulation Control Language" poate fi controlat foarte amănunţit modul de comportare al aplicaţiei în situaţii speciale, care nu fac parte din logica uzuală;
- un aparataj matematic foarte dezvoltat pentru analiza economică a modelului şi a rezultatelor simulării, putându-se astfel deduce o serie de indici de productivitate şi de eficienţă.

Toate programele din pachetul QUEST dispun de o interfaţă grafică foarte evoluată, permiţând modelarea 3D a obiectelor utilizate cu ajutorul unui modul CAD încorporat. De asemenea pot fi utilizate obiectele aflate în bibliotecile foarte bogate ale programului sau pot fi importate modele din majoritatea aplicaţiilor CAD/CAM aflate în utilizare curentă, cum ar fi Pro/Engineer, CaDDy, Autocad[c]. In Figura 2.5-3 se poate observa un model realizat cu ajutorul programului QUEST[d] .

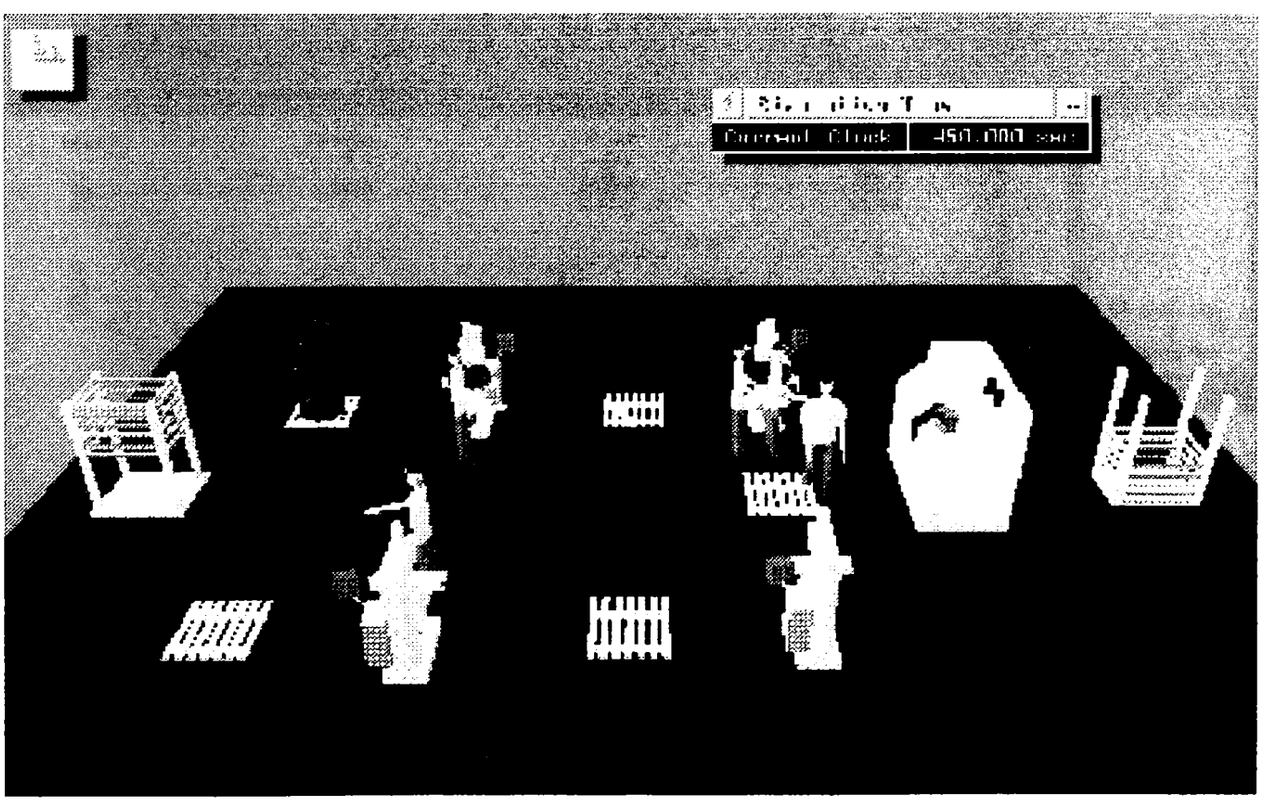


Figura 2.5-3

[c] Pro/Engineer este proprietatea firmei "Parametric Technology" USA, CADdy ese proprietatea firmei "Ziegler Informatics" Germania iar Autocad este proprietate firmei "Autodesk" USA

[d] Modelul a fost realizat în scop didactic la Kuopio Institute of Technology, Finlanda

2.5.a.b. Utilizarea programului SIMUL8

Tot în categoria programelor de simulare bazate pe grafică animată se include şi programul Simul8[e]. Programul este scris pentru mediul Windows, fiind de mai mică amploare şi deci mai puţin pretenţios la nivelul resurselor hard ale sistemului de calcul. Animaţia ce însoţeşte simularea este realizată într-un mediu 2D, folosind obiecte grafice foarte simple, importate din biblioteci sau create ad-hoc cu ajutorul unui editor la nivel de bitmap. O caracteristică interesantă a acestui program o constituie interfaţa cu programul Excel/VisualBasic[f] prin intermediul cărora se poate conduce operaţia de simulare cu ajutorul unor parametrii furnizaţi pe parcursul rulării în ferestre de dialog. Un alt atribut al acestei aplicaţii îl constituie posibilitatea de urmărire a performanţelor sistemului prin intermediul unor grafice specifice fiecărui obiect din mediul grafic.

In Figura 2.5-6 este redată fereastra grafică a programului cu ceasul de simulare şi cu câteva grafice şi diagrame reprezentând parametrii simulării pentru câteva obiecte din model.

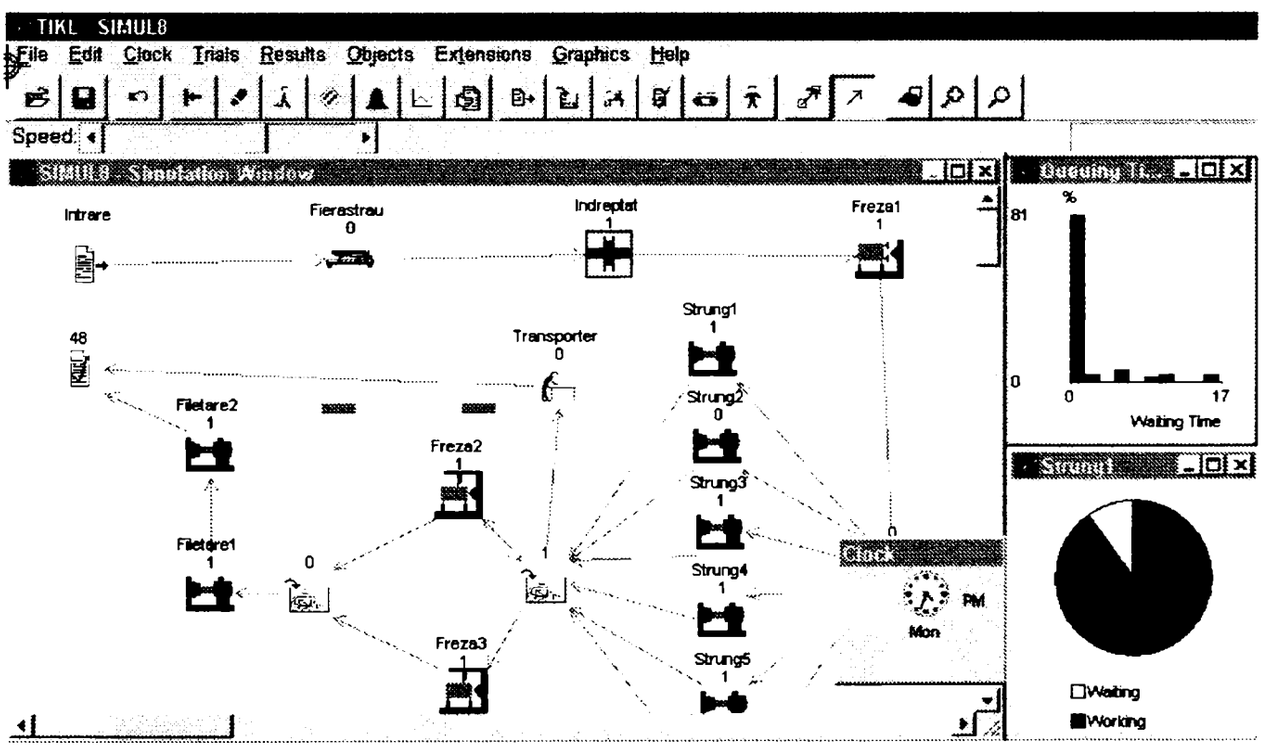


Figura 2.5-4

Programul foloseşte obiecte individuale de tipul "work center", "work entry point", "work exit point", "conveyor", precum şi obiecte de tipul "resources" care se comportă ca resurse partajate între celelalte tipuri de obiecte (operatori umani, scule, sisteme de transport, dispozitive diverse, etc.). Piesele care circulă în sistem sunt configurate iniţial într-o bază de

[e] Simul8 este proprietatea firmei Visual Thinking International Limited, URL: www.VisualT.com
[f] Programele Excel şi VisualBasic sunt proprietatea firmei Microsoft

date sub numele de "work item". Acestora li se asociază imagini grafice simple care sunt folosite în timpul desfăşurării animaţiei. Pentru a realiza simularea, inclusiv animaţia corespunzătoare la nivel de grafică 2D, se folosesc proprietăţile pieselor definite anterior, sub numele de "attributes". Acestea se modifică pe parcursul desfăşurării simulării în conformitate cu logica de desfăşurare a procesului.

2.5.a.c. Utilizarea programului Taylor II

Un program de modelare-simulare deosebit de elaborat pentru folosirea pe platforme PC este programul Taylor II[g]. Acest program permite utilizarea unei interfeţe grafice până la nivel 3D, în funcţie de puterea de redare grafică a echipamentului de calcul. Pe această interfaţă se pot dispune obiecte foarte diverse preluate dintr-o bibliotecă deosebit de bogată, sau se pot crea obiecte noi cu ajutorul unui editor de imagini. Fiecărui obiect i se pot conferi proprietăţi de animaţie şi un anumit comportament în timpul rulării programului. De asemenea se poate defini o logică de rutare a pieselor în sistem pe baza unor ferestre de dialog sau pe baza unor extensii de cod care sunt scrise într-un limbaj special denumit TLI care face parte din grupa limbajelor orientate pe obiecte.

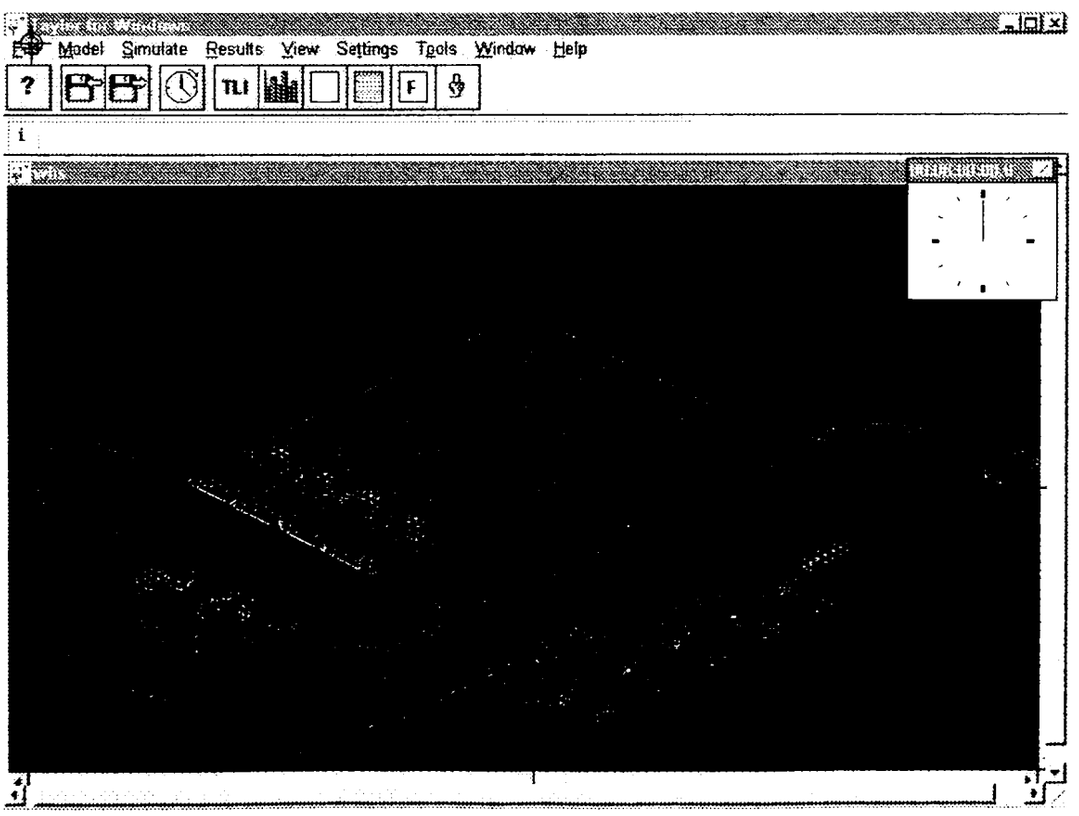


Figura 2.5-5

[g] Taylor II este proprietatea firmei F&H Simulations Inc. URL: http:\\www.taylorii.com

Figura anterioară redă fereastra principală a aplicației, care conține spre exemplificare un model al unei magazii automatizate. Se pot observa meniurile aplicației precum şi ceasul care redă scurgerea timpului de simulare. Fiecare obiect prezent în model are posibilitatea de a fi configurat prin precizarea unor proprietăți, fie prin intermediul unei ferestre specializate, fie prin intermediul unor secvențe de cod. O astfel de fereastra de configurare este prezentată în Figura 2.5-6. Obiectul configurat este un buffer dispus în fața magaziei, prin intermediul căruia sunt transferate piesele în/din conveior dinspre/înspre alimentatorul matricial care deserveşte magazia.

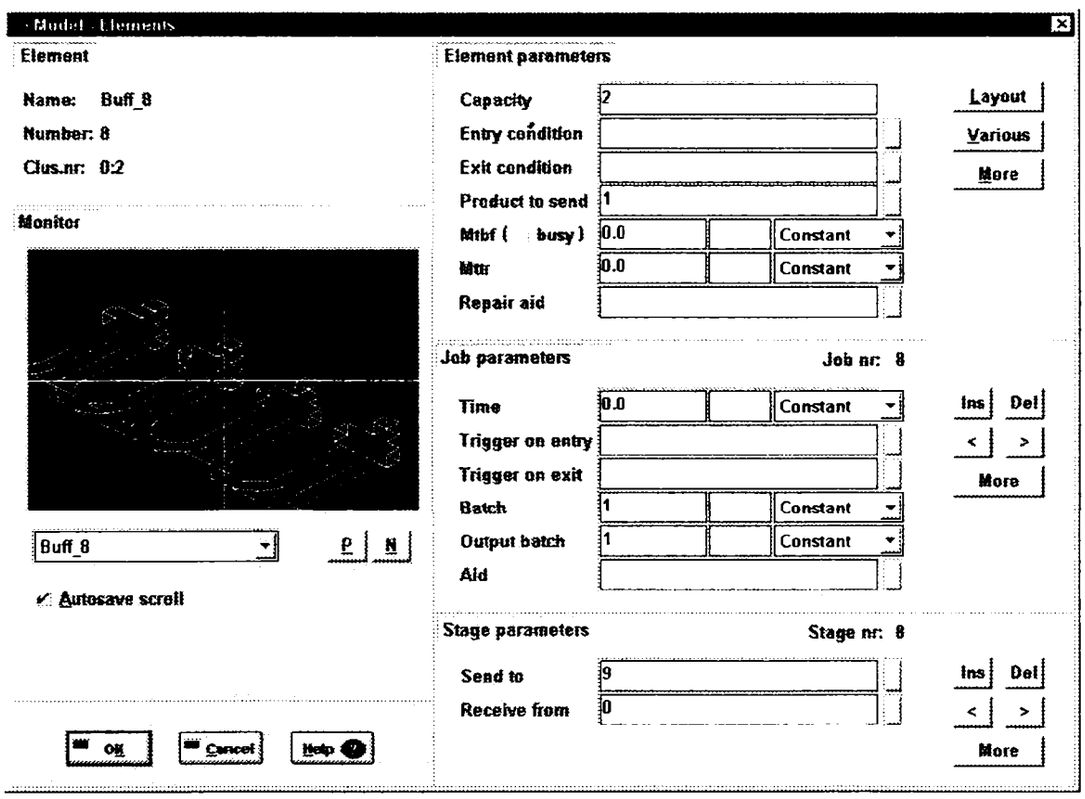


Figura 2.5-6

Din analiza figurii de mai sus se poate observa că fereastra permite configurare simultană a trei categorii de parametrii:

- parametrii specifici elementului selectat (capacitate, condiții de intrare/ieşire, etc.);
- parametrii specifici sarcinii de lucru ce se desfăşoară In postul respectiv (timpul de lucru, mărimea lotului, timpii de întrerupere accidentală pentru reparații, etc.);
- parametrii de legătură cu celelalte componente ale modelului (legături de intrare, legături de ieşire).

2.5.b Modelarea şi simularea cu ajutorul tehnicilor simbolice

2.5.b.a. Utilizarea programului DesignCPN pentru simularea cu reţele Petri colorate

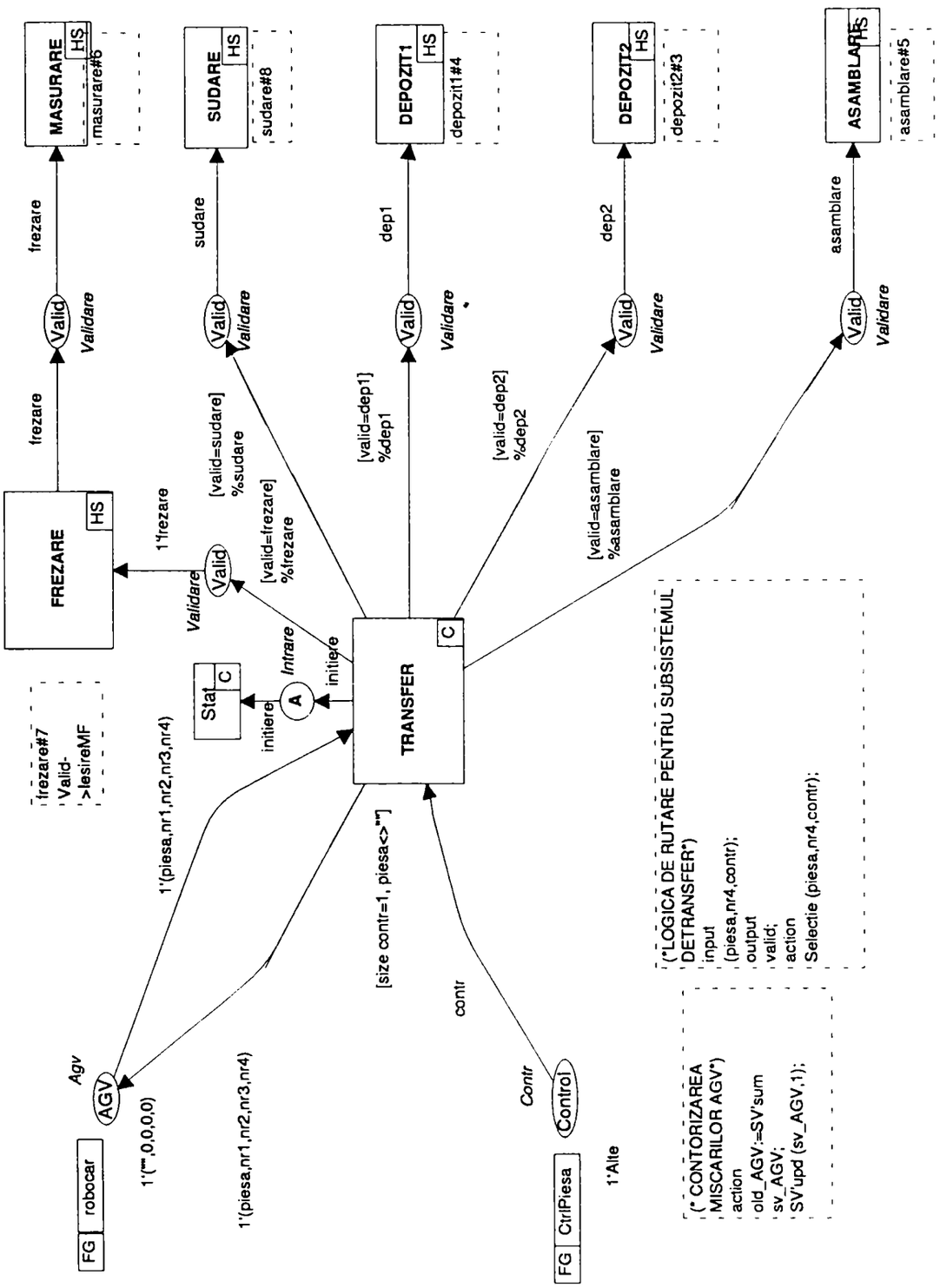


Figura 2.5-7

Metoda de modelare-simulare bazată pe utilizarea reţelelor Petri a debutat ca o metodă matematică, bazată pe calcule matriciale. Ideea de simulare se materializează prin transferarea

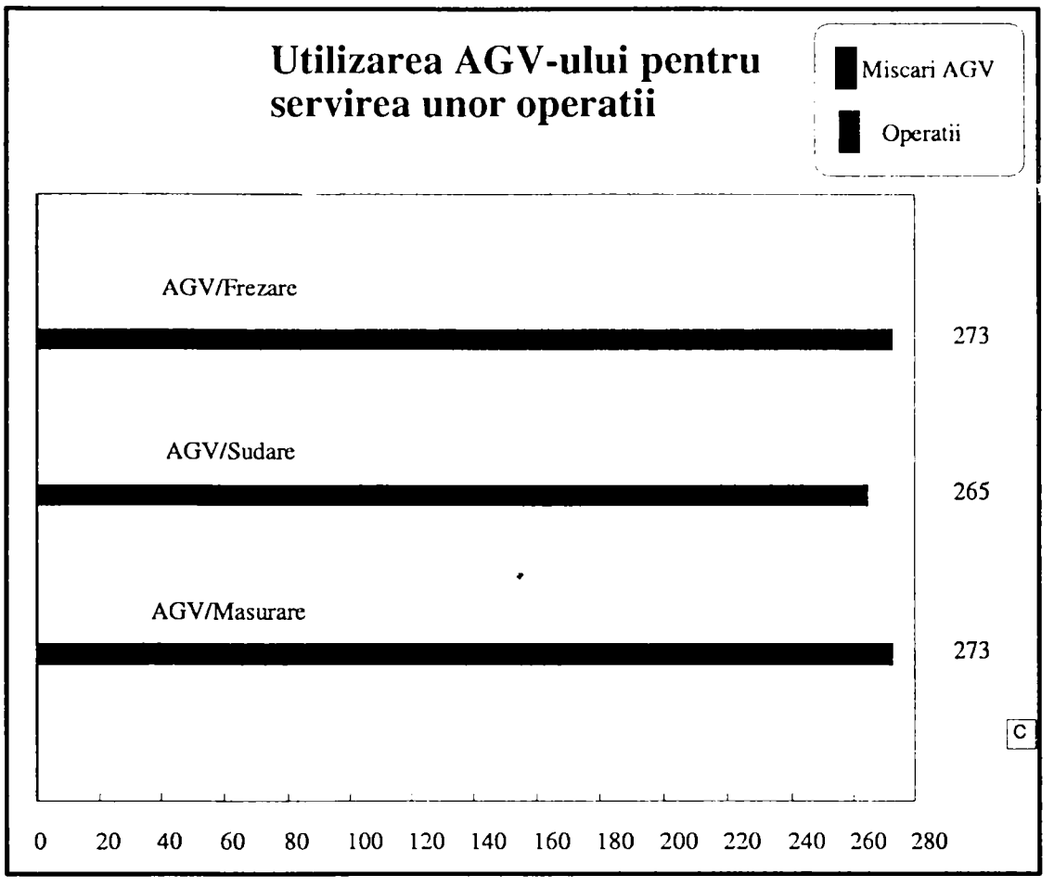
unor jetoane dintr-un loc în altul într-o rețea imaginară, pe baza unor reguli bine precizate. Puterea de calcul a computerelor actuale face posibilă vizualizarea mişcării acestor jetoane într-un mediu grafic ce are trei componente principale: poziții, tranziții şi arce.

Implementarea cunoscută sub numele de DesignCPN se bazează pe utilizarea teoriei rețelelor Petri colorate, care sunt "animate" cu ajutorul unui limbaj special din grupa limbajelor orientate pe obiecte denumit Meta Language[h]. Acest limbaj se bazează pe o rețea de poziții, tranziții şi arce realizată cu ajutorul unui editor grafic într-un mediu X-Windows[i] specific sistemelor de operare Unix. Figura 2.5-7 prezintă o pagină dintr-o diagramă mai complexă realizată cu acest program, folosită şi mai departe în prezenta lucrare pentru susținerea unor afirmații teoretice. Pe baza unor diagrame de acest fel se realizează întreaga *logică de "funcționare" a modelului*, care trebuie să reproducă cu maximă fidelitate comportamentul real al sistemului. Pentru a realiza acest lucru, programatorul are la dispoziție următoarele elemente:

- *un set de culori*, care reprezintă diversele clase de "obiecte" ale modelului (piese, dispozitive, condiții de validare, proprietăți, stări, etc.);
- *un set de variabile*, care aparțin claselor anterioare şi care reprezintă conținutul momentan al acestora;
- *inscripțiile de pe arce*, cu ajutorul cărora sunt controlate condițiile de validare ale tranzițiilor;
- *condițiile de gardă ale tranzițiilor*, care sunt condiții booleene care permit sau inhibă validarea unor tranziții;
- *secvențele de cod*, asociate tranzițiilor, prin intermediul cărora sunt controlate valorile variabilelor de ieşire sau sunt efectuate diverse alte operații specifice limbajului ML (actualizarea unor variabile statistice, actualizarea unor reprezentări grafice, calculul unor parametrii de eficiență, etc.).

Sistemul permite şi reprezentarea grafică a unor parametrii de stare, actualizați pe baza secvențelor de cod asociate tranzițiilor. Din multitudinea de reprezentări grafice posibile, în Figura 2.5-8 este redat un exemplu din care se poate deduce numărul de mişcări ale unui AGV în comparație cu numărul operațiilor de frezare, sudare şi asamblare desfăşurate în sistem.

[h] Meta Language este proprietatea lui Meta Software Corporation, Cambridge MA, URL: www.metasoft.com
[i] X-Windows este proprietatea lui Massachusetts Institute of Technology




```
init
SV'init sv_AGV;
SV'init sv_Fre;
SV'init sv_Sud;
SV'init sv_Mas;

action
 SC_upd_chart{
  sc = "Bar Chart 1",
  values = [
   (1, [!old_AGV+1, !old_Fre+1], true),
   (2, [!old_AGV+1, !old_Sud+1], true),
   (3, [!old_AGV+1, !old_Mas+1], true)]};
```

Figura 2.5-8

2.5.b.b. Programul VisualObjectNet++ pentru simularea cu ajutorul rețelelor Petri

Programul, a cărui interfață grafică este prezentată în Figura 2.5-9, face parte din categoria simulatoarelor cu rețele Petri. Ceea ce îl distinge de programul DesignCPN prezentat anterior, este faptul că *pot fi modelate şi transformări continui*, descrise de funcții matematice sau ecuații diferențiale, alături de transformările discrete. O diagramă construită cu acest program poate cuprinde o varietate mai mare de fenomene dar are dezavantajul de a

folosi rețele binare, cu limitările binecunoscute, provenind din imposibilitatea de a dispune de seturi complexe de jetoane într-o anumită poziție la un moment dat.

Programul a fost scris în C++ şi poate fi compilat pentru diverse platforme de lucru. Varianta utilizată în unele părți ale lucrării de față (doar pentru exemplificare) a fost rulată sub WindowsNT 4.0.[j] Autorul programului este Dipl.-Ing. Rainer Drath[k] de la Universitatea Tehnică din Ilmenau, Germania.

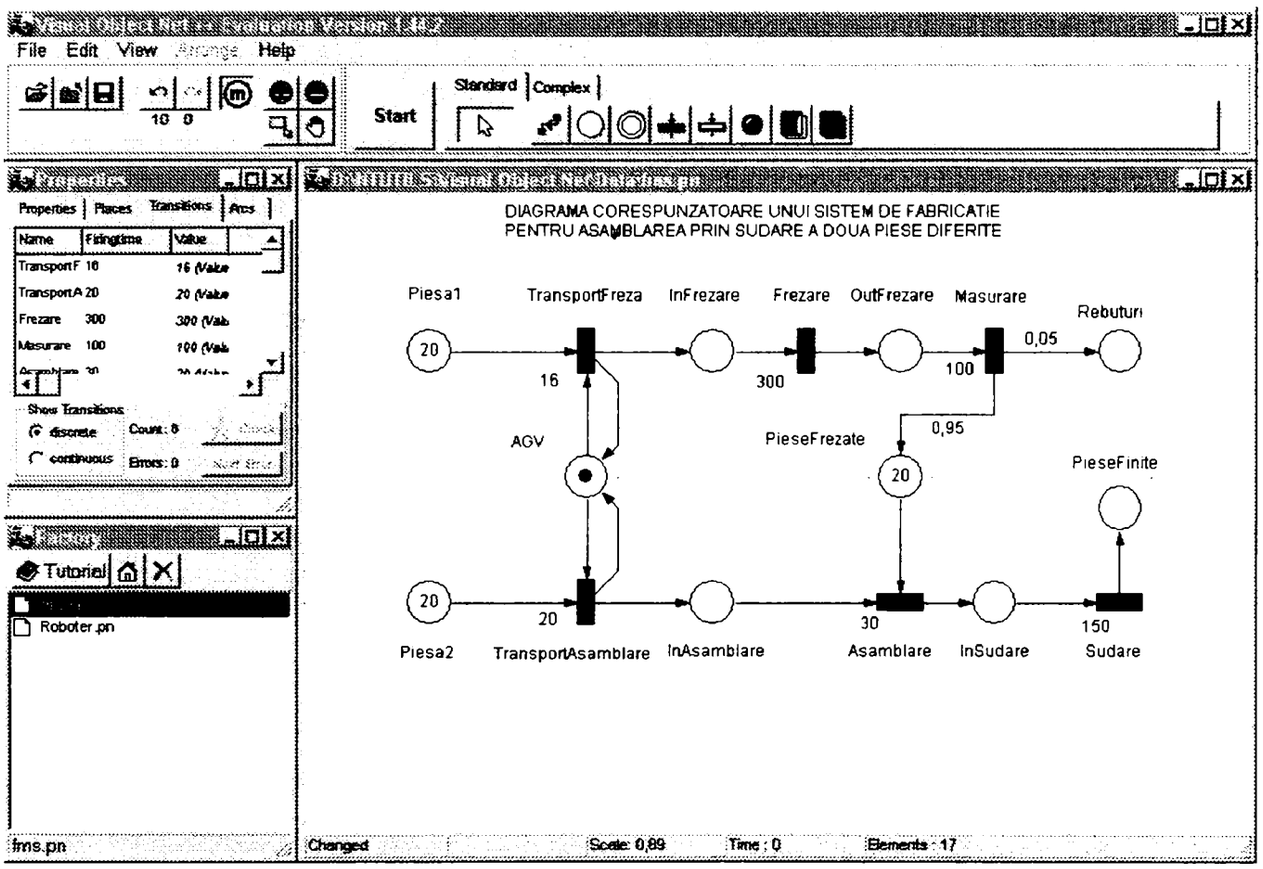


Figura 2.5-9

După cum se poate observa din figura de mai sus, programul oferă o interfață grafică pe care se poate desena diagrama şi se pot preciza în ferestre text independente diferite detalii legate de construcția şi funcționarea acesteia. Meniul principal şi butoanele rapide ale aplicației sunt dispuse în partea superioară a ecranului. In partea stângă, sub meniul principal, este dispusă o altă fereastră, în care se introduc parametrii caracteristici ai fiecărui obiect din diagramă. De asemenea mai există încă o fereastră tot în partea stângă, care permite gestionarea aplicațiilor la nivel de fişiere, încărcarea unei diagrame, lansarea secțiunilor de help ş.a.m.d. Interfața grafică a aplicației permite salvarea diagramei în mai multe formate grafice, inclusiv formate post script.

[j] WindowsNT 4.0. este proprietatea firmei Microsoft.
[k] URL: http:\\www.systemtechnik.tu-ilmenau.de\~drath\

2.6. Arhitectura soft a unei celule autonome de fabricație

Celulele autonome de fabricație sunt componentele de bază ale unui sistem de fabricație flexibil, autonom şi descentralizat [Michiko Matsuda, Kuniko Inoue, 1991, pag. 173]. O asemenea celulă autonomă de fabricație impune trei funcții de bază:

- selecția unei sarcini de fabricație;
- execuția sarcinii de fabricație;
- raportarea rezultatului după execuția sarcinii.

Mai multe celule de fabricație autonome sunt conectate la o bază de date comună şi la un depozit comun de materiale. *Fluxul de materiale din depozit corespunde fluxului de informații din producție*. Celula autonomă primeşte din magazie un material care are ataşate informații despre procesul de producție şi îl returnează în magazie cu o anumită operație executată şi astfel cu un alt conținut informațional. Conducerea centralizată a sistemului flexibil prin intermediul unui computer central de sistem introduce o oarecare limitare a flexibilității, prin ierarhizarea prea accentuată a fluxului de informații. Descentralizarea informației şi concentrarea ei la nivelul celulei deschide noi perspective procesului de organizare a sistemelor flexibile.

In cele ce urmează va fi prezentat un *model de conducere centralizată ierarhică* a unui sistem flexibil. Un controler de nivel superior furnizează sarcini de producție pentru componentele de nivel inferior, sub formă de seturi de date sau sub formă de secvențe de program. Informația se transmite în mod rigid de la nivelul superior spre cel inferior, unde se execută instrucțiunile de bază de prelucrare a semifabricatului. Schema de organizare a unui asemenea flux de informații organizat pe principii ierarhice este prezentată în Figura 2.6-1. Informațiile de producție sunt divizate în *informații despre produs* şi *informații despre proces*.

Informațiile despre produs se referă la produsul propriu-zis şi ele sunt furnizate din etapa de proiectare a produsului.

Informațiile despre proces sunt divizate la rândul lor în:

- date de nivel sarcină de producție;
- date de nivel secție de producție;
- date de nivel bază de cunoştințe de producție.

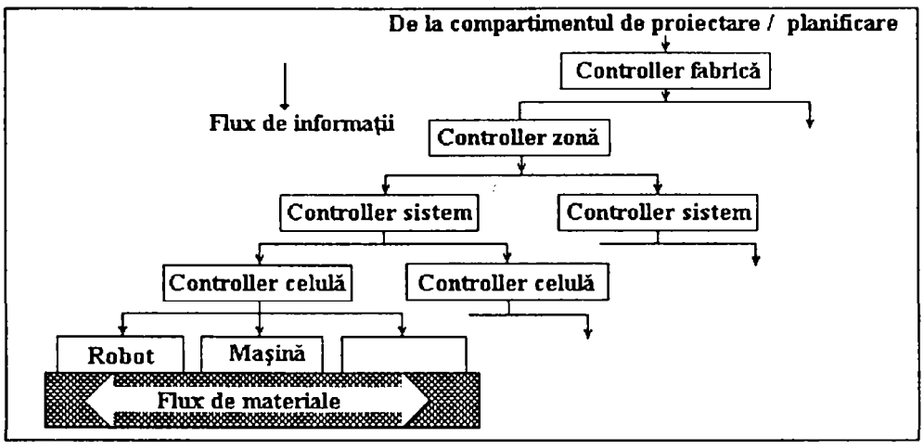


Figura 2.6-1

Astfel *datele la nivel de sarcină de producţie (production task)* se vor referi la procedurile de lucru. In orice celulă de fabricaţie execuţia unui program de fabricaţie duce la mişcări ale diverselor componente ale celulei, cum ar fi mişcări ale roboţilor, ale centrelor de prelucrare de strunjire, frezare sau altele, etc. Datele la nivel de sarcină organizează de fapt toate aceste mişcări într-o anumită succesiune, impusă de programul de fabricaţie.

Datele la nivel de secţie constau din informaţii dinamice sau statice despre structura celulelor autonome, tipurile de maşini utilizate, parametrii tehnologici nominali ai acestora, precum şi datele provenite din inspecţia tehnică periodică a acestora.

Bazele de cunoştinţe de producţie sunt conţinute de regulă în sisteme expert, care pot fi utilizate atât la elaborarea unor programe de fabricaţie optimale, cât şi la depanarea sistemului. Legăturile ce se stabilesc pe fluxurile de date dintr-o celulă flexibilă între diferitele categorii de date prezentate anterior sunt redate schematic în Figura 2.6-2.

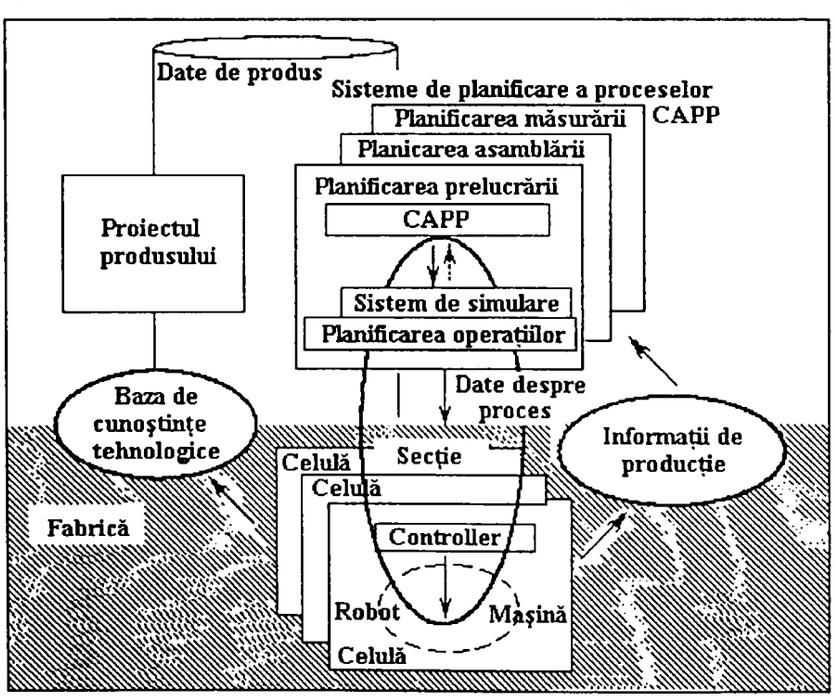


Figura 2.6-2

In cazul unui *sistem cu componente autonome descentralizate*, toate componentele sunt conectate pe acelaşi nivel al fluxului de date, aşa cum se poate observa în Figura 2.6-3. Sistemul de gestiune a producţiei şi celulele de fabricaţie autonome comunică prin intermediul fluxului de informaţii şi de materiale. Celula de fabricaţie autonomă primeşte materialele odată cu informaţii despre produs şi planul de fabricaţie, generate în etapele de proiectare ale acestora. La nivelul celulei sunt elaborate sarcinile de lucru în scopul desfăşurării operaţiilor de prelucrare. După desfăşurarea completă a procesului dintr-o celulă, materialul, care are acum ataşat un alt set de informaţii, va fi promovat în celula următoare şi aşa mai departe, până când la o extremitate a fluxului se va obţine un nou produs. In acest mod fiecare celulă a sistemului flexibil va lucra independent dar cooperant. Sistemul de fabricaţie autonom nu este controlat de un computer central unic aşa cum este îndeobşte cazul la un sistem flexibil clasic. Dimpotrivă, acest sistem va fi construit din celule individuale independente, angajate într-un proces permanent de cooperare.

Particularităţile de compunere şi organizare a fabricaţiei în beneficiul ideii de autonomie, la nivel de celulă sau de sistem, se pot grupa în două categorii:

- particularităţi legate de funcţionarea celulei;
- particularităţi legate de organizarea bazelor de date şi a fluxului de informaţii care să suporte funcţiunile de bază ale celulelor autonome.

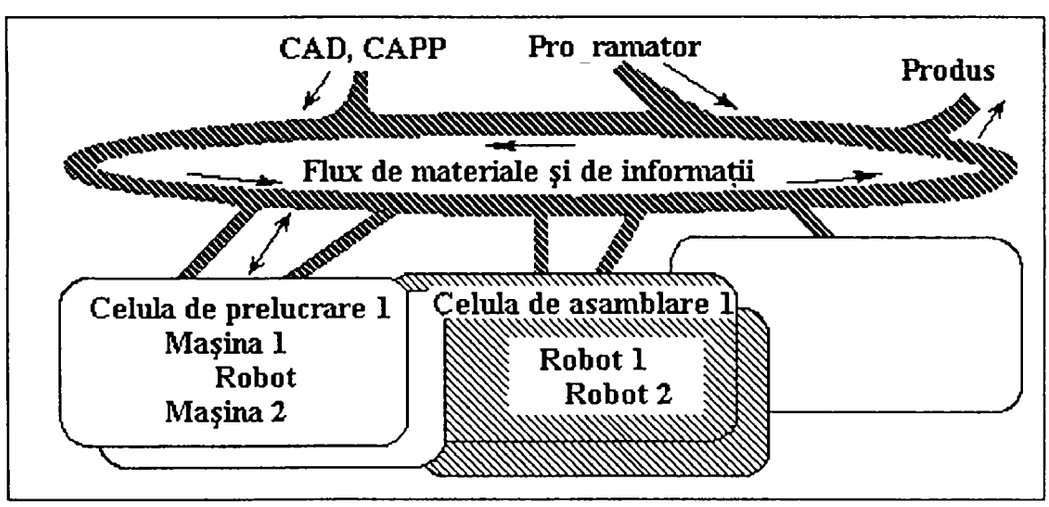


Figura 2.6-3

Particularităţile legate de funcţionarea celulei se manifestă la toate tipurile de celule autonome, fie că ele sunt de prelucrare, de măsurare, sau de asamblare. Aşa după cum se observă în Figura 2.6-4, *funcţiile de conducere obligatorii ale unei celule autonome* sunt următoarele:

- *selecția unei sarcini de prelucrare executabile*. La nivelul celulei se cunoaşte tipul şi amploarea sarcinilor de prelucrare ce se pot executa. La apariția unui reper în spațiul celulei aceasta îşi testează capacitățile şi starea, precum şi nivelul de prioritate al operației cerute. Dacă operația se încadrează în limitele celulei şi aceasta este vacantă sau are în lucru un reper cu nivel de prioritate mai scăzut, atunci piesa este reținută în celulă pentru prelucrare.
- *generarea de comenzi* pentru execuția unei sarcini de prelucrare. La nivelul celulei sunt elaborate comenzile concrete necesare precum şi seturile de date pentru execuția unui reper: itinerar tehnologic, comenzi de control, programe NC, programe de supraveghere a prelucrării.
- *raportarea rezultatului* operațiilor desfăşurate. Celula de prelucrare validează rezultatul acțiunilor sale precum şi stadiul în care a ajuns prelucrarea reperului. In raportul furnizat de celulă se găsesc date despre acuratețea prelucrării, lista operațiilor care au fost executate, lista operațiilor carte au mai rămas de executat, sau cel puțin operația în mod necesar următoare.

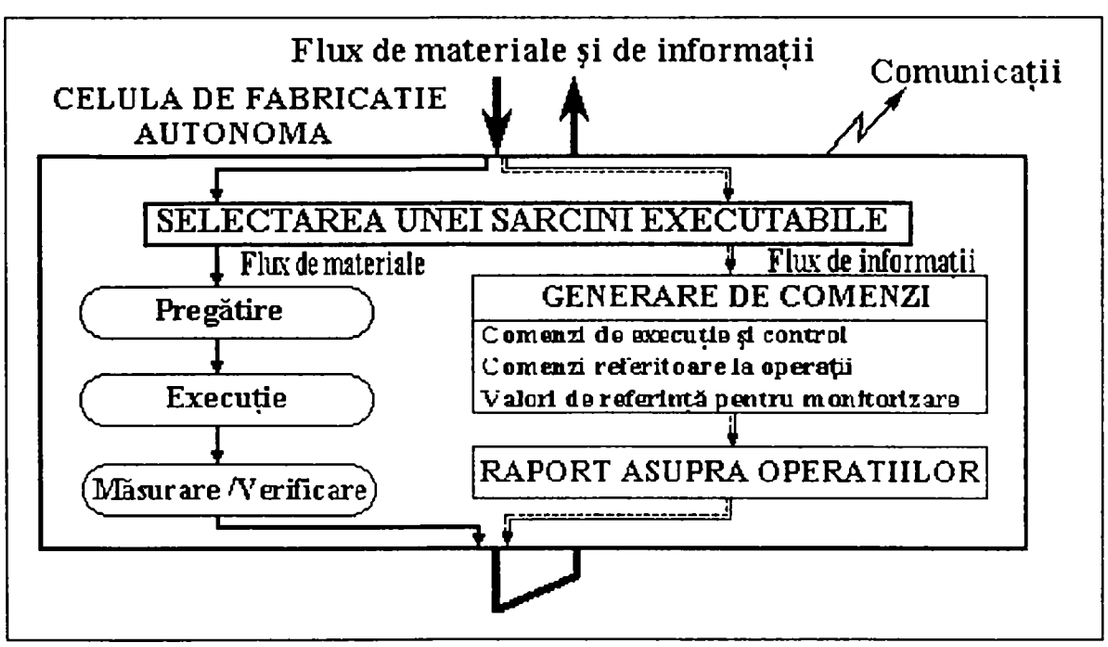


Figura 2.6-4

Pe lângă aceste trei funcții de producție, este nevoie de o *funcție de comunicație*, care să organizeze aceste celule autonome descentralizate.

Bazele de date şi gestiunea lor sunt atribute proprii ale celulei autonome. Aceste baze de date conțin date de producție a celulei, incluzând structura acesteia, specificații despre maşini, reguli de translație de la comenzi virtuale la comenzi fizice, reguli de organizare,

reguli de alegere a sarcinilor. Câteva tipuri de date şi modalităţi de gestionare a lor sunt prezentate în Figura 2.6-5.

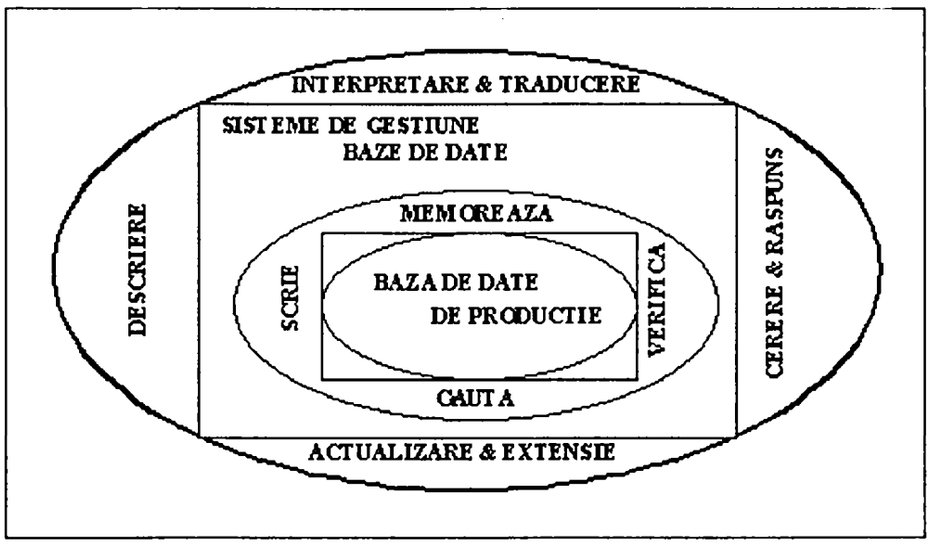


Figura 2.6-5

Principalele activităţi care se desfăşoară în cadrul gestiunii bazelor de date sunt:

- *cerere şi răspuns*. Serviciul de răspuns stabileşte care dintre maşini este capabilă să îndeplinească o anumită sarcină de lucru, sau care este starea celulei la un moment dat. Pentru a răspunde la o anumită cerere, este necesar nu numai să existe accesată baza de date, dar şi să se ruleze anumite programe de simulare pentru a vedea care este răspunsul optim al sistemului.
- *interpretare şi traducere*. In cadrul acestei atribuţii se desfăşoară un proces de integrare în contextul real a comenzilor virtuale lansate în sistem, iar apoi o traducere a acestora în comenzi executabile în conjunctura respectivă, pe baza cărora sistemul poate emite ordine de mişcare.
- *actualizare şi extensie*. Acest tip de serviciu permite sistemului să-şi lărgească conţinutul cu noi componente sau structuri şi să-şi dezvolte baza de date de producţie în consecinţă. In plus, serviciul determină noi relaţii între sistemele de date şi permite actualizarea periodică a informaţiei provenite de la senzori în condiţiile reale de lucru.

In cele ce urmează se va prezenta pe scurt un sistem de fabricaţie structurat pe celule de fabricaţie flexibile autonome. In Figura 2.6-6 sunt redate principalele structuri şi procese dintr-un sistem de fabricaţie bazat pe celule autonome de prelucrare. Fiecare celulă autonomă este conectată la depozitul de informaţii şi de materiale. Prin materiale se înţeleg nu numai materiile prime, semifabricatele sau piesele finite, ci şi scule, dispozitive, palete, etc. Fiecare material conţine informaţii despre el însuşi şi poate răspunde la întrebări legate de procesele

care mai sunt de executat asupra lui, care este starea momentană, care va fi forma finală. Informaţiile din baza de date corespund cu materialele din depozit. Materialele curg în sistem ataşate la datele de produs şi planificarea producţiei. O celulă de prelucrare autonomă ia materialul din magazie cu un anumit conţinut de informaţii şi îl depune înapoi cu informaţiile actualizate. Circulaţia materialelor în magazie este controlată de o celulă specială de monitorizare a informaţiei în depozit.

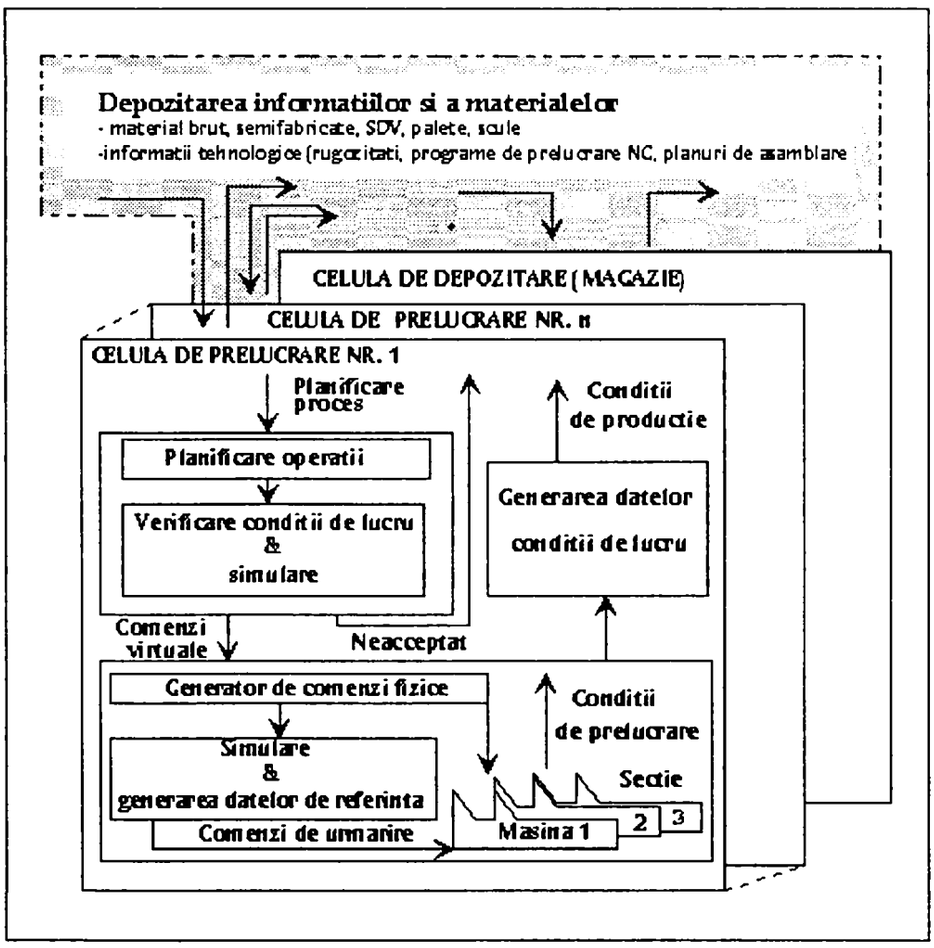


Figura 2.6-6

O celulă de fabricaţie autonomă procesează operaţiile pe baza informaţiilor conţinute în magazie. Procesul care se desfăşoară într-o celulă autonomă este descris în continuare. Baza de date a celulei şi serviciile pe care aceasta le poate asigura sunt elemente de care se va ţine cont în întregul proces. *Acţiunile principale desfăşurate în sistem* sunt următoarele:

- celula *distribuie* o anumită sarcină la una din maşinile componente în concordanţă cu un plan de producţie acceptat în prealabil. După aceasta, celula creează un plan de operaţii pentru fiecare maşină, verifică starea sarcinilor existente la momentul respectiv în celulă şi decide asupra acceptării sarcinii de prelucrare folosind simularea. Dacă sarcina este acceptată, procesul înaintează spre pasul următor;

- pe baza planului de operații, celula *generează comenzi* pentru sistemul de comandă numerică, robot, etc. In plus, celula generează *criterii de urmărire* pe bază de simulare;
- rezultatul şi starea după o anumită operație este *tradus în informație necesară în pasul următor* în procesul de prelucrare. Celula returnează în magazie materialul prelucrat împreună cu acest set de informații.

Arhitectura soft a sistemelor bazate pe celule autonome include trei nivele principale, aşa după cum se poate urmări în Figura 2.6-7:

- bazele de date şi serviciile din celulă;
- nivelele de descriere în fluxul de informații;
- simularea.

Inima sistemului soft prezentat în figura de mai sus este o *bază de date de producție* cu servicii speciale de prelucrare a datelor-răspuns, interpretare, actualizare. Fiecare celulă are propria ei bază de date de producție. Intrările şi ieşirile soft la nivelul unei celule autonome sunt informațiile despre produs şi planul de producție. Conținutul informațiilor referitoare la produs se modifică de-a lungul itinerarului de prelucrare.

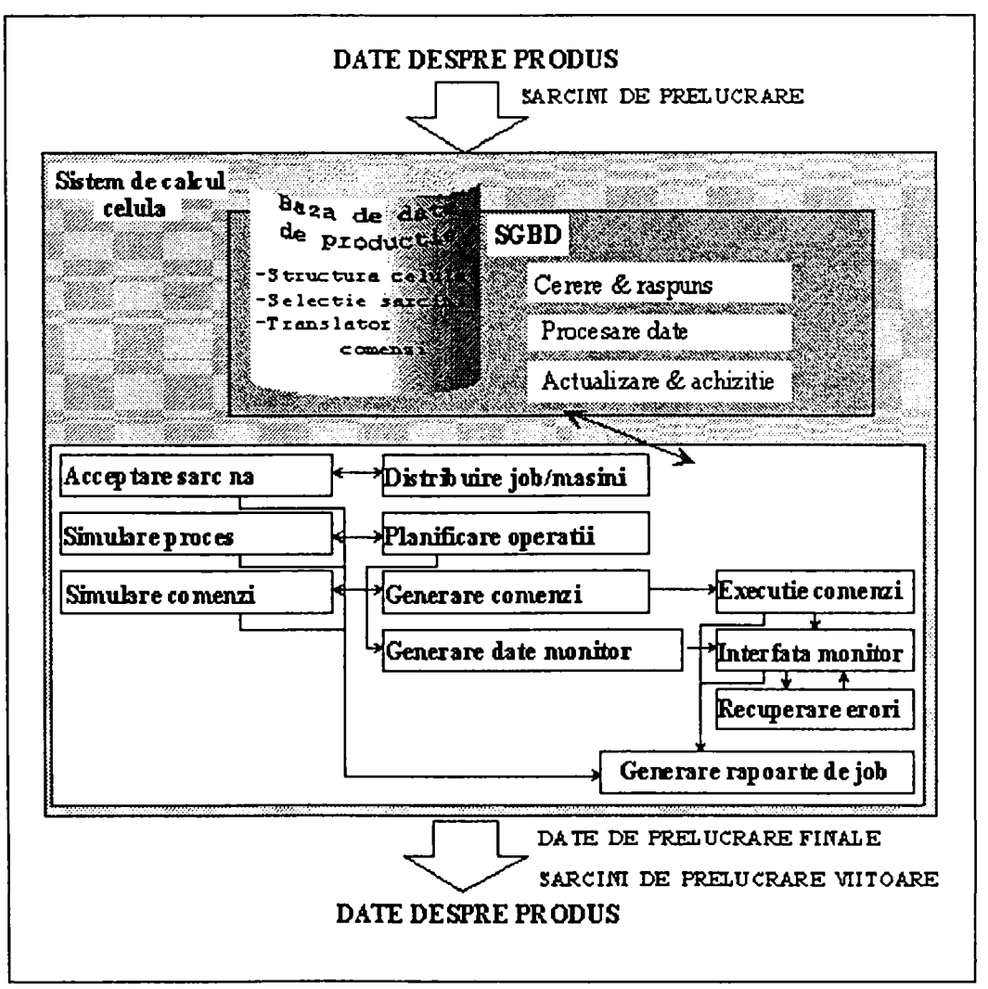


Figura 2.6-7

Serviciul de acceptare a sarcinilor ține cont de rezultatele furnizate de serviciile de distribuire a sarcinilor, planificare a operațiilor şi simulare a proceselor. Serviciul de distribuire a sarcinilor foloseşte serviciul de răspuns al bazei de date. Serviciile de planificare a operațiilor şi de simulare a procesului de fabricație se referă de asemenea la serviciile de răspuns ale bazelor de date de producție. Serviciul de generare a comenzilor foloseşte serviciile de interpretare care pot transforma comenzile virtuale în comenzi specifice unei anumite maşini unelte particulare.

Serviciul de generare a datelor de supraveghere (monitorizare) produce seturi de date folosind simulatorul de comenzi precum şi datele prelevate la nivelul secției de producție.

Funcția sau *serviciul de execuție* include compilatoarele de comenzi şi monitorizează starea maşinilor unelte şi a centrelor de prelucrare. Dacă în timpul lucrului apare o eroare, se activează *serviciul de recuperare în caz de eroare*. Astfel este apelat serviciul de simulare a comenzilor şi baza de cunoştințe folosită la generarea acestora. Dacă celula nu a mai întâlnit o asemenea eroare, baza de date va fi actualizată cu acest nou set de cunoştințe. Serviciul de generare a rapoartelor de execuție furnizează pe baza datelor preluate din proces rapoarte structurate în funcție de cerințele impuse de etapele ulterioare de lucru asupra aceluiaşi reper.

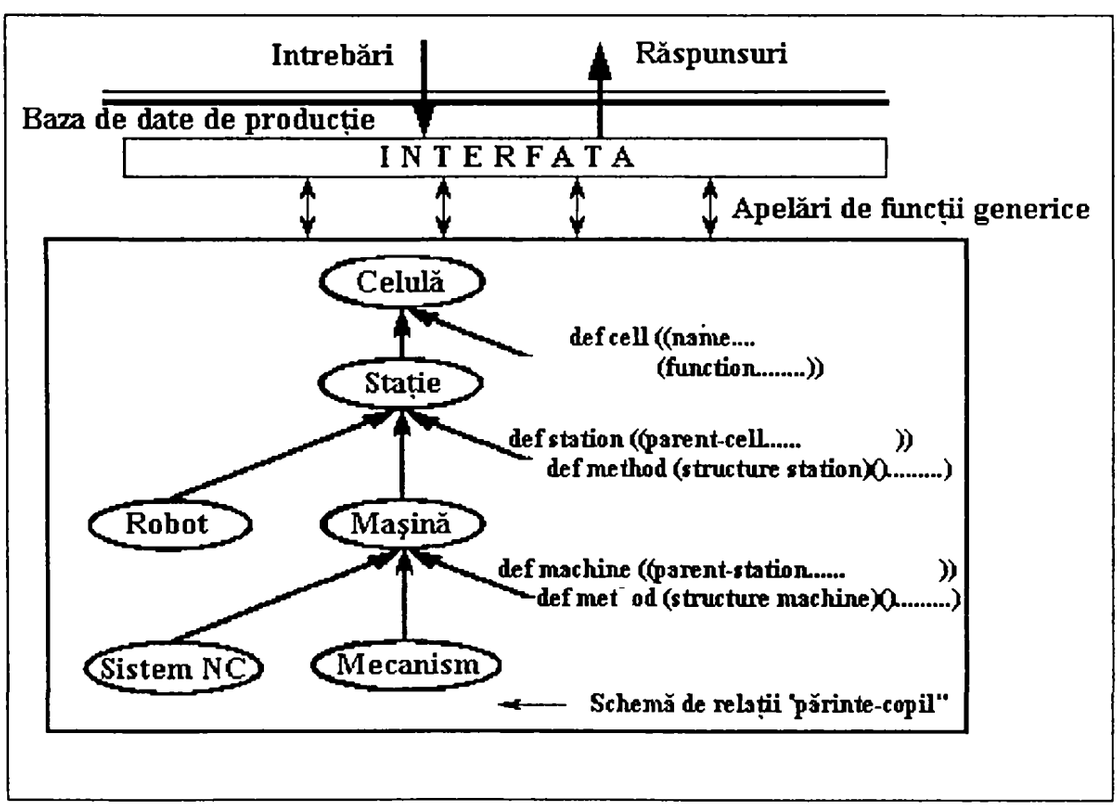


Figura 2.6-8

Bazele de date de producție şi serviciile orientate pe obiecte sunt componentele esențiale ale unei arhitecturi soft de celulă independentă. In ultimul timp sistemele de gestiune ale bazelor de date au numeroase facilități de programare orientată pe obiecte. Avantajele acestei abordări constau în posibilitățile foarte diverse de descriere a informației şi în posibilitățile de manipulare a unei mari varietăți de tipuri de date. Datele pot fi asociate cu obiecte complexe între care pot fi realizate relații nemijlocite. In Figura 2.6-8 este reprezentată organizarea unei celule experimentale pentru care a fost scris un fragment de program folosind limbajul Flavor, o variantă a limbajului Lisp. [în **** Symbolycs Inc., 1988, pag. 368-490].

```
*structure cell
CELL 1
STATION 1
HG500
(MECHANISM1, NC1)
6PPL
STATION2
HG650
(MECHANISM2, NC2)
8PPL
*show device
HG500_____STATION1
6PPL_____STATION1
HG630_____OFF
8PPL_____STATION2
HG650_____STATION2
*change HG500 HG630
STATION1
*structure cell
CELL1
STATION1
6PPL
HG630
(MECHANISM1, NC1)
STATION2
8PPL
HG650
(MECHANISM2, NC2)
```

Figura 2.6-9

Celula se compune din unități de prelucrare, descompuse la rândul lor în centre de prelucrare, roboți, etc. Sub acest nivel se pot distinge mecanisme, unități de control CNC, etc. Fiecare element al celulei constituie un obiect, între acestea putându-se stabili relații diverse. Atributele unui obiect sunt constituite din caracteristicile sale şi toate celelalte elemente de descriere ale acestuia. Chiar şi apartenența lui la o anumită categorie superioară constituie un atribut. Fiecare element de pe un nivel inferior se poate raporta la elementul situat imediat deasupra lui. Elementele de pe un anumit nivel nu au în general informații despre obiectele desfăşurate pe nivelele inferioare. Structura celulei poate fi construită prin lansarea unei rutine de întrebare-răspuns de sus în jos, obținându-se astfel întreaga ierarhie a celulei, cu toate atributele elementelor componente. In acest fel înlocuirea unui anumit obiect din structură cu un altul echivalent presupune doar declararea corectă a atributelor noului obiect, fără să fie

necesară notificarea expresă a acestei manevre. Câteva manevre tipice de tipul întrebare-răspuns sunt prezentate în fragmentul de program cuprins în Figura 2.6-9. Prima întrebare, marcată cu o săgeată, se referă la structura celulei 1. In răspuns se afirmă că stația 1 este compusă din centrul de prelucrare HG500 şi dintr-un transportor de palete 6PLL.

Următoarea problemă este legată de starea de moment a componentelor din celula 1. Se pot astfel remarca stațiile 1 şi 2 formate dintr-un centru de prelucrare şi un transportor de palete, precum şi un centru de prelucrare denumit HG630 care nu este conectat. O altă comandă este cea de schimbare a utilizării centrelor de prelucrare HG500 şi HG630 prin asocierea lor la alte celule de prelucrare. In urma acestei manevre la următoarea comandă de afişare a structurii celulei, în cadrul celulei 1 va fi afişat centrul de prelucrare HG630.

Fluxurile de informații şi limbajele de programare au drept obiect datele de producție şi planificarea proceselor. Sistemele de planificare a proceselor generează planuri de proces pentru celulele virtuale. De exemplu un sistem de planificare a proceselor poate genera un plan de prelucrare de genul: 1. frezare contur A, 2. găurire orificiu B, etc. Prin intermediul unor programe speciale aceste comenzi sunt transformate în altele mult mai detaliate, care sunt adecvate centrului de prelucrare pe care se va executa operația respectivă, comenzi care se referă la tipul de scule utilizate, regimurile de aşchiere, generarea traiectoriilor de mişcare a sculelor, etc. Intr-o etapă ulterioară aceste comenzi vor fi traduse într-un limbaj specific echipamentelor de comandă numerică cu care sunt echipate centrele de prelucrare. Dacă operațiile sunt executate, va fi generat un raport asupra tuturor acestor operații. Dacă acestea nu se execută, atunci materialul va fi returnat în magazie împreună cu planul de operații. Acest proces de transformare a caracterului informației este prezentat în Figura 2.6-10.

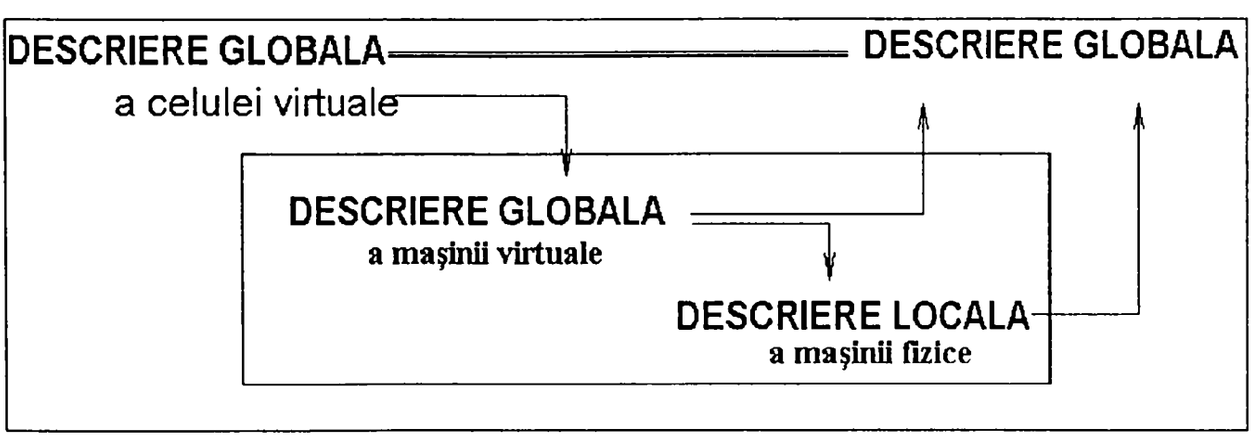


Figura 2.6-10

In concluzie, se impune elaborarea unui protocol standard pentru comanda celulei virtuale şi pentru descrierea relaţiilor dintre obiectele care o compun, implementat cu ajutorul unui limbaj orientat pe obiecte. In urma implementării acestui limbaj de programare se vor situa toate celelalte proceduri de comandă concretă a tuturor componentelor celulei, folosind de această dată limbaje proprii şi comenzi specifice fiecărui obiect.

Funcţia de simulare stă la baza oricărui proces de decizie într-o celulă de fabricaţie autonomă. Abilitatea şi calitatea simulării au un efect major asupra nivelului de autonomie al unei celule de fabricaţie. Tehnicile de simulare se pot referi la simularea programului de fabricaţie, la simularea proceselor sau la simularea comenzilor.

2.7. Bibliografie

1. **** Symbolycs Inc., 1988, Symbolics Common Lisp-Language Concepts, #999055.
2. Abel, D., 1990, Petri-Netze für Ingenieure. (Springer-Verlag, Berlin).
3. Alla, H., Ladet, P., 1986, Coloured Petri Nets: A Tool for Modelling, Validation and Simulation of FMS, in A. Kusiak (Editor), Flexible Manufacturing Systems, Methods and Studies, Elsevier Science Publishers B. V. (North - Holland).
4. David, R., Alla, H., 1994, Petri Nets for Modeling of Dynamic Systems- a survey. Automatica, 30, 175-202.
5. Douglas Lyon, 1996, On the Use of Stochastic Petri Nets for the Performance of Real-time N^{th} Order Stochastic Composition, *www.bridgeport.edu/subpage*.
6. Francisc Kovacs, Sanda Grigorescu, Cornel Rădulescu, 1994, Sisteme de fabricaţie flexibilă robotizate, partea a II-a, Litografia UPT, Timişoara.
7. Hauke Jungnitz, Alan Desrochers, 1991, An Overview of Analytical Performance Analysis Models for Manufacturing Systems, in Design, Analysis and Control of Manufacturing Cells, ASME, PED- Vol. 53.
8. Hietiko, E., 1996, Computer Aided System for Preliminary Design of Screen Cylinder Variants. C Technica 90, Acta Universitatis Ouluensis Technica, Department of Mechanical Engineering, Oulu.
9. Hrúz, B., 1995, Control of Discrete Event Dynamic Systems in the Context with Flexible Manufacturing Systems and Petri Nets. Report 100, Helsinki University of Technology, Control Engineering Laboratory.
10. Kaplan, S., Keter, A., Epstain, S., A., ş.a., 1990, COPILOT-A PC-Based Expert System for Reactor Operational Assistance Using a Bayesian Diagnostic Module. Reliability Engineering and System Safety 30, 219-237.
11. Katz, R., 1994, Contemporary Logic Design. Benjamin/Cummings Publishing Company, Inc., New York.
12. L'Ecuyer, P., Blouin, F., Couture, R., 1993, A Search for Good Multiple Recursive Random Number Generators. ACM Transactions on modeling and Computer Simulation. 3(2):87-98.
13. Michiko Matsuda, Kuniko Inoue, 1991, Software Architecture of Autonomous Manufacturing Cells, ASME, PED -vol. 53, Design Analysis and Control of Manufacturing Cells.
14. O'Keefe M., R., 1989, The Role of Artificial Intelligence in Discrete-Event Simulation. Artificial Intelligence, Simulation and Modeling (edited by Laurence E. Widman ş.a., Jonh Wiley & Sons, New York, 1989).
15. Peterson, J., 1977, Petri Nets. ACM Computing Surveys. 9(3):223-252.
16. Rembold, U., Dillman, R., 1986, Computer-Aided Design and Manufacturing. Methods and Tools. Springer-Verlag, Berlin.
17. Ruiz-Mier, S., Talavage, J., 1989, A Hybrid Paradigm for Modeling of Complex Systems. Artificial Intelligence, Simulation and Modeling (edited by Laurence E. Widman ş.a., Jonh Wiley & Sons, New York, 1989).
18. Rumbaugh, J., Blaha, M., ş. a., 1991, Object-Oriented Modeling and Design. (Prentice-Hall International, Inc.).
19. Shu, H., 1990, A Prototype Expert System for Pressure Vessel Design. Linköping Studies in Science and Technology, Linköping University, 95p.
20. Viswanadham, N., Narahari, Y., 1992, Performance Modeling of Automated Manufacturing. (Prentice-Hall, Englewood Cliffs, New Jersey).

3. UTILIZAREA REȚELELOR PETRI LA ANALIZA PERFORMANȚELOR TEHNOLOGICE ALE SISTEMELOR FLEXIBILE DE FABRICAȚIE

3.1. Cuprins

Sistemele flexibile de fabricaţie fac parte în esenţă din categoria *sistemelor cu evenimente discrete*, la care evoluţia în timp a sistemului depinde de interacţiunea diverselor evenimente separate, cum ar fi alimentarea cu semifabricate a unei maşini, ieşirea din sistem a unei piese finite, defecţiunile apărute la unele maşini, etc. In evoluţia unui astfel de sistem se pot identifica situaţii în care *unele subsisteme prezintă un comportament continuu în timp*. De exemplu pe durata prelucrării unei piese, deşi starea sistemului general nu se modifică, la nivelul subsistemlui de prelucrare se produc transformări continui a masei de material sau de reziduri, a conţinutului informaţional al piesei rezultate, etc. Starea unui astfel de sistem se modifică însă esenţial doar la anumite momente şi nu continuu în timp, aşa cum se întâmplă de exemplu la un sistem de transfer de căldură sau de curgere a unor fluide. Modelarea funcţionării sistemelor cu evenimente discrete are două aspecte distincte:

- *modelarea calitativă*, care urmăreşte aspecte cum ar fi controlabilitatea sistemului, stabilitatea, existenţa punctelor de blocare;
- *modelarea cantitativă*, având drept obiective determinarea unor parametrii cum ar fi productivitatea sistemului, cantitatea de produse aflată în proces, precum şi alţi parametrii de ordin cantitativ.

După cum este prezentat în [Viswanadham, N., Narahari, Y., 1992, pag. 3 şi urm.], principalele probleme la care trebuie să răspundă modelarea calitativă sunt:

- care este probabilitatea ca un produs să fie livrat înainte de un anumit termen;
- care este numărul minim de maşini de lucru necesare pentru ca productivitatea medie a sistemului să atingă o anumită valoare;
- câte dispozitive de prindere şi palete sunt necesare pentru ca indicele de utilizare al utilajelor să depăşească 80%;
- capacitatea de producţie este suficientă pentru a produce şi livra o anumită cantitate de produse până la un anumit termen;
- care este "lay-out"-ul care oferă cea mai mare flexibilitate;
- care este numărul minim de resurse din sistem (maşini, dispozitive de transfer, magazii intermediare) care asigură o probabilitate de minim 95% pentru producerea unui anumit număr de repere în condiţiile prezenţei întreruperilor accidentale ale maşinilor;
- care este efectul blocării unei maşini asupra productivităţii sistemului şi asupra timpului principal de prelucrare;
- care sunt punctele înguste în sistem ;
- daca sistemul prezintă blocaje şi care este timpul mediu până la apariţia acestora;

- care sunt influențele numărului de componente de un anumit tip din sistem asupra productivității acestuia şi a timpului principal de fabricație;
- dacă există capacități disponibile în sistem pentru preluarea unor sarcini de fabricație de prioritate mai redusa.

3.2. Principalii parametrii cantitativi folosiți în analiza sistemelor de fabricație

In continuare vor fi prezentați parametrii principali de analiză cantitativă (modelare cantitativă) pentru un sistem de fabricație [Viswanadham, N., Narahari, Y., 1992, pag. 37 şi urm.].

3.2.a Timpul principal de fabricație (TPF)

Timpul principal de fabricație *(TPF) este timpul total necesar pentru obținerea unui produs la nivelul unei întreprinderi sau sistem de fabricație.*

Timpul total de fabricație *(TTF) este timpul scurs de la comandarea materiilor prime până la livrarea produsului finit către beneficiar.*

In mod ideal, timpul principal de fabricație ar trebui să fie egal cu suma timpilor de prelucrare şi de asamblare. Aceasta poate fi valabil dacă lotul de fabricație este de mărime 1, nu există timpi de manipulare a materialelor, timpi de pregătire, timpi de întrerupere sau de defectare a utilajelor, etc. Fireşte că astfel de condiții ideale nu pot fi întâlnite în practică. De aceea vor fi prezentate în continuare principalele componente ale TPF, împreună cu factorii principali care le influențează.

3.2.a.a. Timpul de pregătire

Timpul de pregătire *este timpul necesar pentru o maşină sau pentru un sistem de fabricație pentru a trece de la un produs la altul.*

In categoria timpilor de pregătire se include timpii necesari pentru prindere-desprindere, schimbarea sculelor şi pregătirea locului de muncă. Reducerea timpilor de pregătire are un profund efect asupra flexibilității unui sistem de fabricație, constituind baza filozofiei de producție "just in time". Timpul de pregătire se manifestă de fiecare dată când un nou lot de produse este încărcat pe sistem. *Mărimea economică a lotului de fabricație "Q"* este acel număr de repere pentru care suma cheltuielilor de pregătire şi a celor de inventar este minimă.

Costurile totale pe unitate de produs din lot, "C", sunt date de ecuația (3.2-1):

$$C = \frac{R}{Q}S + \frac{Q}{2}I \qquad \textbf{(3.2-1)}$$

unde:

S = suma costurilor de pregătire pe lot;

R = rata de apariție a unui anumit reper în lot;

I = costurile de inventar pe reper.

Prin derivarea acestei ecuații în raport cu "Q" şi formularea condiției de minim "dC/dQ=0" se poate obține mărimea optimă a lotului "Q" pentru care costurile sunt minime, conform relației (3.2-2) :

$$Q = \sqrt{\frac{2SR}{I}} \qquad \textbf{(3.2-2)}$$

Se observă din analiza acestor relații că diminuarea timpilor de pregătire de "n" ori va conduce la o diminuare a lotului optim de $\sqrt{n}$ ori, deci implicit o reducere a costurilor de inventar.

3.2.a.b. Timpul principal de prelucrare

Timpul principal de prelucrare *este timpul necesar pentru schimbarea proprietăților fizico-chimice ale semifabricatului la un anumit post.*

Acesta este singurul interval de timp în care se schimbă valoarea produsului. Toate celelalte intervale de timp nu modifică valoarea semifabricatului şi trebuiesc minimizate. Relațiile de calcul ale timpului principal de prelucrare sunt specifice diverselor procedee tehnologice.

3.2.a.c. Timpul de manipulare (de transfer)

Timpul de transfer *este considerat timpul folosit pentru deplasarea pieselor dintr-un loc de muncă la altul.*

Manevrarea (manipularea, transferul, transportul) piesei se poate face în interiorul spațiului de lucru al maşinii, în interiorul unei secții, al unei întreprinderi, sau între diverşi subcontractanți care participă la realizarea unui anumit produs. Evident că raționalizarea timpilor de manevră constă în reducerea lor la valori minimale şi suprapunerea cu timpii de maşină. Există în principal trei căi de reducere a acestor timpi de manevră:

- *utilizarea unor centre de prelucrare evoluate*, care dispun de magazii de scule foarte cuprinzătoare, capabile să execute un număr mare de sarcini de prelucrare fără manevrarea pieselor pe alte maşini;
- organizarea producţiei pe *principiul tehnologiilor de grup*, astfel încât toate maşinile unelte necesare pentru realizarea unei anumite familii de repere să fie grupate şi aliniate într-o succesiune logică;
- utilizarea unor *sisteme de transfer de mare eficienţă*, cum ar fi transportoarele cu lanţuri, AGV-uri inteligente, încărcătoare frontale şi multe altele.

3.2.a.d. Timpul de aşteptare

Timpul de aşteptare *este intervalul de timp petrecut de un reper în aşteptarea eliberării unei resurse care să-l transfere în etapa următoare a procesului de fabricaţie.*

Aceste resurse sunt de cele mai multe ori centrele de prelucrare sau sistemele de transfer care sunt partajate de întregul sistem de fabricaţie şi care au de îndeplinit simultan mai multe sarcini de producţie.

3.2.b Numărul de piese în lucru (PIL)

Numărul de piese în lucru *(PIL) reprezintă cantitatea de repere în diverse faze de fabricaţie aflate la un moment dat în sistem.*

In accepţiunea clasică, numărul de piese în lucru era o garanţie pentru trecerea cu succes a unor momente de întrerupere în funcţionare sau în alimentarea cu semifabricate. Intr-o accepţiune mai modernă, numărul mare de piese în lucru înseamnă o reducere a flexibilităţii şi o creştere a timpilor de aşteptare a reperelor în vecinătatea utilajelor. Acestea sunt aspecte care trădează vicii de organizare şi care sunt, deci, nedorite.

PIL se poate aprecia prin produsul dintre rata de transfer a liniei (sau a celulei) şi timpul necesar pentru ca un reper să iasă de pe linie, altfel spus TPF.

In mod ideal, PIL ar trebui să fie egal cu numărul de maşini al celulei, astfel încât nici o maşină să nu aştepte vreun semifabricat iar acestea să nu se acumuleze în aşteptarea eliberării vreunui utilaj.

3.2.c Rata de fabricaţie (RF)

Pentru un sistem de fabricaţie, *rata de fabricaţie* se exprimă prin numărul de repere sau piese obţinute într-o unitate de timp, oră sau zi. Inversul ratei de fabricaţie este intervalul

de timp necesar producerii unui reper. Pornind de la această definiție, în cazul unei linii de transfer tehnologic rata de fabricație va fi inversul tactului liniei (unități de timp pe reper). In cazul unor celule flexibile, relația de calcul pentru rata de fabricație este mai dificil de obținut. In această situație, simularea pe baza unui model fidel poate oferi valori convenabile pentru acest parametru.

3.2.d Capacitatea de fabricație (CF)

In cazul unor celule de fabricație *capacitatea de fabricație* se poate exprima prin numărul de repere ce pot fi produse pe linia respectivă într-un anumit interval de timp de mai mare amploare (numărul de repere pe schimb sau pe 24 de ore, de exemplu). Acest parametru nu se obține prin simpla multiplicare a ratei de fabricație cu timpul de referință ci se ține cont şi de aspectele de fiabilitate a celulei, cu caracter statistic. Uneori este dificil de apreciat capacitatea unei celule de fabricație sau a unei întreprinderi printr-un număr de repere produse, mai ales dacă acestea nu au un caracter omogen. De aceea se pot folosi şi alți parametrii, cum ar fi *timpul total de maşină* pe un interval de referință.

3.2.e Flexibilitatea

Un *sistem flexibil* este un sistem capabil să răspundă la schimbări, iar *flexibilitatea* este proprietatea unui sistem de a răspunde efectiv la schimbare.
Schimbările care pot afecta un sistem pot avea proveniență internă (defecțiuni ale echipamentelor, variații ale timpului de aşchiere pe diverse maşini, absenteismul personalului angajat, probleme de calitate) sau proveniență externă (schimbări ale documentației, modificări ale cerințelor de piață). Noțiunea de flexibilitate este strâns legată de cea de *competitivitate*. Puterea de a răspunde prompt cerințelor pieței prin produse noi, de bună calitate şi cu timpi de realizare cât mai reduşi, atribut ce se leagă de flexibilitate, face ca o întreprindere sau o celulă de fabricație să fie competitivă din punct de vedere economic şi tehnic.

3.2.f Performabilitatea

Aprecierea performanțelor unui sistem trebuie să includă şi aspectele probabilistice legate de timpii de întrerupere datorați defecțiunilor de utilaj. Din acest punct de vedere, se utilizează două noțiuni distincte, ambele legate de caracterul mai mult sau mai puțin performant al unui sistem: fiabilitatea şi disponibilitatea.

Fiabilitatea F(t) a unui sistem este probabilitatea "P" ca sistemul să lucreze continuu pe un interval de timp [0,t] şi să fabrice produse corespunzătoare calitativ.

Dacă se notează cu "T" variabila aleatoare care reprezintă timpul de întrerupere din sistem, atunci definiția fiabilității pe un interval de timp [0,t] se poate exprima prin relația:

$$F(t) = P\{T < t\}$$

(3.2-3)

Este evident faptul că valoarea fiabilității depinde *de distribuția cumulată a timpului de întrerupere*, notată cu "C(t)":

$$F(t) = 1 - C(t)$$

(3.2-4)

Se notează cu "f(t)" densitatea de probabilitate a lui "T". Cu aceste notații, timpul mediu până la apariția unei defecțiuni (TMD) se va determina cu relația:

$$TMD = \int_0^{\infty} tf(t)dt$$

(3.2-5)

Dacă definim variabila aleatoare "T_R" ca fiind timpul de reparații, iar cu "f(t)" densitatea de probabilitate a acestei variabile, atunci se va putea calcula timpul mediu de reparație (TMR) cu relația:

$$TMR = \int_0^{\infty} tf(t)dt$$

(3.2-6)

Disponibilitatea "D" a unui sistem la un moment "t" este considerată a fi probabilitatea ca sistemul să se afle în funcțiune la momentul "t". Dacă nu există defecțiuni, este evident că disponibilitatea sistemului va fi 1, deci el va fi în funcțiune în orice moment. Se poate demonstra că disponibilitatea unui sistem de fabricație este:

$$D = \frac{TMD}{TMD + TMR}$$

(3.2-7)

3.2.g Calitatea

Satisfacerea pretențiilor de calitate ale partenerilor de fabricație, fie că ei sunt subcontractori sau clienți de produse finite, este cel mai înalt deziderat la care trebuie să se supună un proces de fabricație. Dinamica deosebit de accentuată a producției în ultimii ani, impusă de o competiție din ce în ce mai acerbă pe piață, a dus la cristalizarea unui nou

concept de asigurare a calității şi de organizare a controlului de calitate: *principiul calității totale*. Principalele componente ale acestui principiu sunt:

- *controlul în proces*, care presupune organizarea unor puncte de control individual la fiecare post de lucru, unde sunt controlate toate reperele care se produc;
- *calitatea transparentă*, care presupune deschiderea întreprinderilor şi a secțiilor de producție către clienți, care pot aprecia organizarea producției şi indicii de calitate obținuți chiar la fața locului;
- *transferul de autoritate* în domeniul calității la nivelul lucrătorului bine calificat, care are posibilitatea de a opri întreg ciclul de fabricație, dacă constată că nu sunt respectați indicii de calitate;
- *controlul total*, care presupune verificarea întregii producții realizate, bucată cu bucată.

Pe baza acestor principii se poate obține o reducere a costurilor de mentenanță ale produsului, care vor compensa creşterile de costuri de producție, astfel încât, pe ansamblu, sporirea indicilor de calitate ai produselor nu trebuie sa însemne în mod necesar creşterea prețului de producție!

3.3. Prezentarea sistemului de fabricație flexibilă din laboratorul Catedrei OMM a Facultății de Mecanică din Timişoara

O primă abordare din partea autorului a domeniului evaluării cantitative a unui sistem de fabricație pe bază de modelare a fost făcută în lucrarea [Dreucean, M., 1996]. Alături de alte considerații asupra unor aspecte ale modelării sistemelor cu evenimente discrete, în lucrarea amintită sunt prezentate şi definițiile principalilor parametrii cantitativi implicați în acest demers.

Ulterior, pe baza experienței acumulate cu ocazia stagiilor de documentare din străinătate şi a studiilor efectuate, autorul a contribuit la organizarea unui sistem de fabricație automatizată flexibilă la sediul Catedrei de Organe de Maşini şi Mecanisme a Facultății de Mecanică din Timişoara, sistem finanțat într-un program Tempus. Acest sistem a fost folosit în lucrarea de față ca suport real al tuturor metodelor de analiză şi conducere prezentate. Pentru o mai bună înțelegere a etapelor de elaborare a modelului şi a celor de simulare care vor urma, se consideră necesară prezentarea în detaliu a construcției acestui sistem, pornind de la tipurile de piese ce se vor prelucra. La proiectarea şi realizarea acestui sistem a lucrat susținut o echipă condusă de Prof. dr. ing. Francisc Kovacs, echipă din care a făcut parte şi autorul prezentei lucrări.

3.3.a Prezentare generală

Finanțarea proiectului de realizare a sistemului flexibil de fabricație de la sediul Catedrei OMM a fost făcută în cadrul proiectului TEMPUS JEP 3517-92. Acest proiect Tempus a fost orientat spre dezvoltarea învățământului de productică la Facultatea de Mecanică din Timişoara. Amplasamentul sistemului a fost stabilit la parterul corpului de laboratoare din curtea Facultății de Mecanică, unde există o hală de dimensiuni corespunzătoare. Destinația acestui sistem de fabricație este în primul rând didactică şi de cercetare, deşi nu sunt excluse şi întrebuințări în scop direct productiv.

Sistemul a fost realizat prin integrarea unor echipamente existente, într-o structură unitară bazată pe două variante de piese de tip flanşă cu butuc, denumite în prezenta lucrare "Flanşa 1" şi "Flanşa 2". Acestor echipamente li s-au adăugat câteva dispozitive de depozitare şi transfer care au fost proiectate ad hoc.

Principalele operații tehnologice realizate în sistem sunt frezarea, măsurarea şi sudarea. Servirea acestor operații se face de către patru roboți care au atât rol de manipulare a materialelor cât şi de transfer a pieselor pe distanțe scurte, unul din roboți având translația orizontală pe o distanță de 2000 [mm].

Conducerea sistemului este realizată în momentul de față cu ajutorul unor calculatoare PC orientate exclusiv pe controlul axelor robotului, fără posibilități de programare şi compunere a mişcărilor. In acest stadiu se pot executa doar mişcări limitate de capetele de cursă cu confirmările necesare pentru trecerea de la o fază la alta. Nu există un program de conducere integrată a sistemului. Elementele principale pentru realizarea unui astfel de program sunt prezentate în continuarea lucrării de față.

3.3.b Probleme tehnologice

Piesele reprezentative sunt de două tipuri distincte, formând cele două jumătăți ale unui suport de rolă pentru benzi transportoare. Cele două jumătăți se vor îmbina prin sudare. Prima piesă (piesa 1 sau "Flanşa 1" în cele ce urmează) are o gaură centrală şi un şanfren în jurul acesteia destinat îmbinării prin sudare. Desenul acestei piese apare în Figura 3.3-1.
Operațiile tehnologice impuse de piesa 1 sunt următoarele:

- găurire cu burghiu de diametrul ∅20 pe adâncimea de 75 de mm;
- găurire cu burghiu de diametrul ∅50 pe adâncimea de 50 de mm;
- lamare la fundul găurii de ∅50;
- teşirea găurii ∅50 pentru facilitarea asamblării.

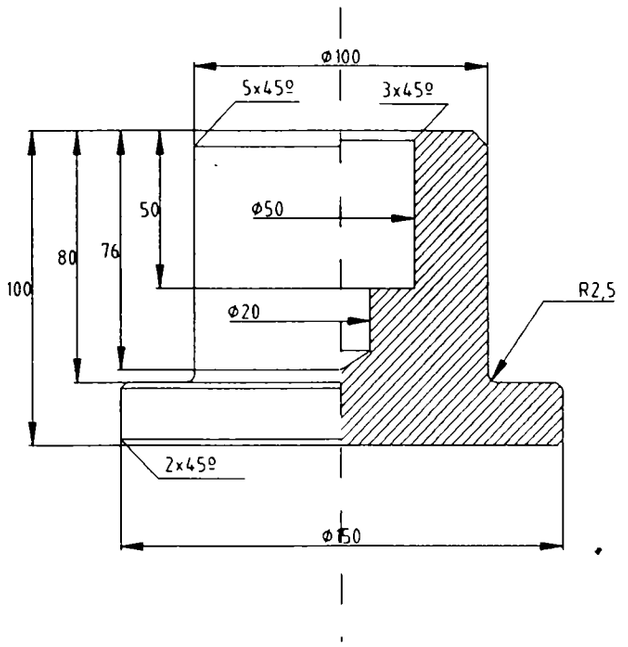


Figura 3.3-1

Pentru a realiza aceste operaţii este nevoie de o maşină de frezat sau de găurit în coordonate. Semifabricatele au forma prezentată în Figura 3.3-3. Acestea nu au nici un detaliu de formă interioară, toate detaliile interne urmând a fi realizate în cadrul sistemului de fabricaţie flexibilă.

A doua piesă, (piesa 2 sau "Flanşa 2"), care se va îmbina cu prima prin sudare, are un cep cilindric la capătul opus flanşei, destinat centrării pe alezajul din contra-piesă. Această piesă nu necesită prelucrări mecanice. Sudarea celor două piese se va realiza cu ajutorul unui robot de sudare de tip Cloos. Desenul acestei piese este redat în Figura 3.3-2.

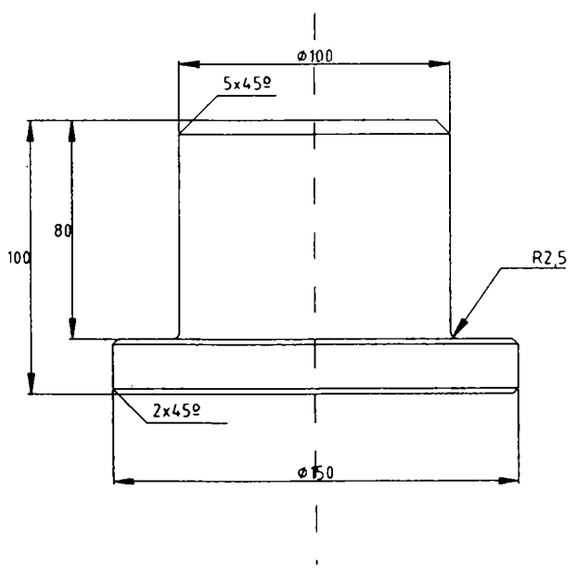


Figura 3.3-3

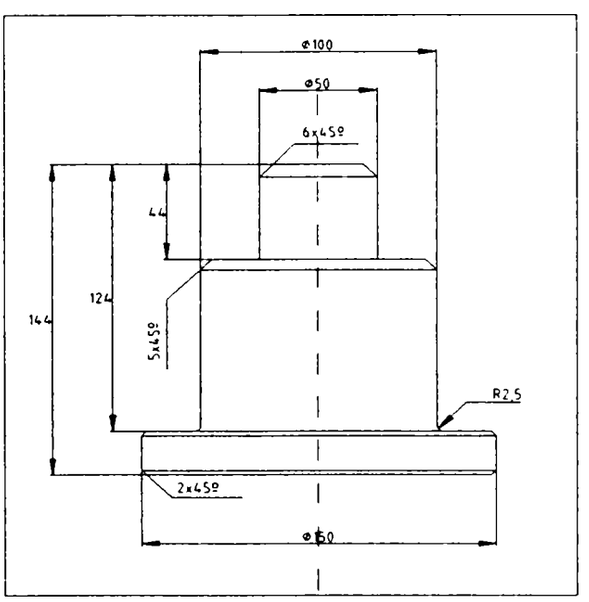


Figura 3.3-2

Făcând un inventar al tuturor operaţiilor tehnologice impuse de piesele reprezentative descrise anterior se poate concluziona că sistemul în care se vor prelucra are nevoie de următoarele maşini de lucru:

- maşină de frezat sau de găurit în coordonate pentru prelucrarea piesei 1;
- instalaţie de sudare cu dispozitiv rotativ;
- instalaţie de măsurare (opţional).

3.3.c Proiectarea planului de amplasament (lay-out)

Proiectarea planului de amplasament a fost făcută pe baza condițiilor existente în hala în care funcționează sistemul şi pornind de la dimensiunile şi spațiile de lucru ale echipamentelor aflate la dispoziție, cu respectarea indicațiilor din [Kovacs, Fr., Grigorescu, S., Rădulescu, C., 1994].

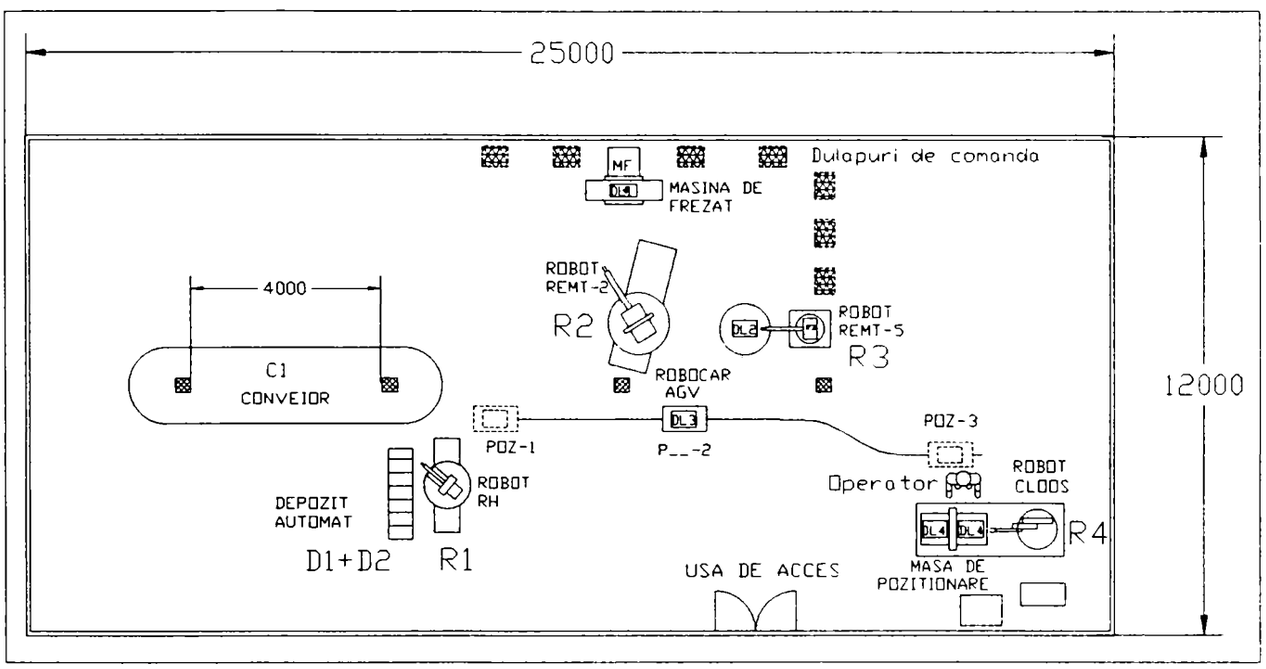


Figura 3.3-4

Planul general de amplasament al sistemului este reprezentat în Figura 3.3-4. Aşa după cum reiese şi din figură, sistemul este format din următoarele componente:

- o maşină de frezat cu cap revolver cu 6 poziții, masă orizontală şi comandă numerică, de tip WMW Heckert. Maşina este prevăzută cu un echipament de comandă numerică iar puterea instalată a maşinii este de 11 KW;
- un robot de sudare fix, de tip Cloos, cu echipament de comandă Cloos şi instalație de sudare MIG;
- un robot de servire a maşinii de frezat cu translație de bază lungă, de tip REMT-2;
- un robot de măsurare de tip REMT-5 care poartă un palpator electronic;
- un robot hidraulic pentru servirea depozitelor D1 şi D2 şi pentru extragerea pieselor de pe conveior;
- un conveior suspendat cu lanț;

- un dispozitiv de transfer la sol de tip vehicul ghidat automat (AGV) care se deplasează rectiliniu pe o traiectorie fixă şi realizează un acces serial al echipamentelor pe care le serveşte;
- un operator uman care serveşte robotul de sudare pentru schimbarea pieselor şi încărcarea/descărcarea lor de pe dispozitivul de transfer.

Operaţiile distincte care se pot desfăşura în acest sistem sunt următoarele:

- alimentare de pe conveior;
- frezare;
- verificare (măsurare);
- asamblare;
- sudare;
- transfer la sol;
- manipulare
- depozitare.

Pentru realizarea acestor operaţii de bază s-au proiectat şi realizat *patru celule de fabricaţie independente*:

- celula de gestionare a conveiorului şi a depozitelor;
- celula de frezare
- celula de măsurare;
- celula de sudare.

In Figura 3.3-5 este reprezentat un detaliu al celulei de fabricaţie.

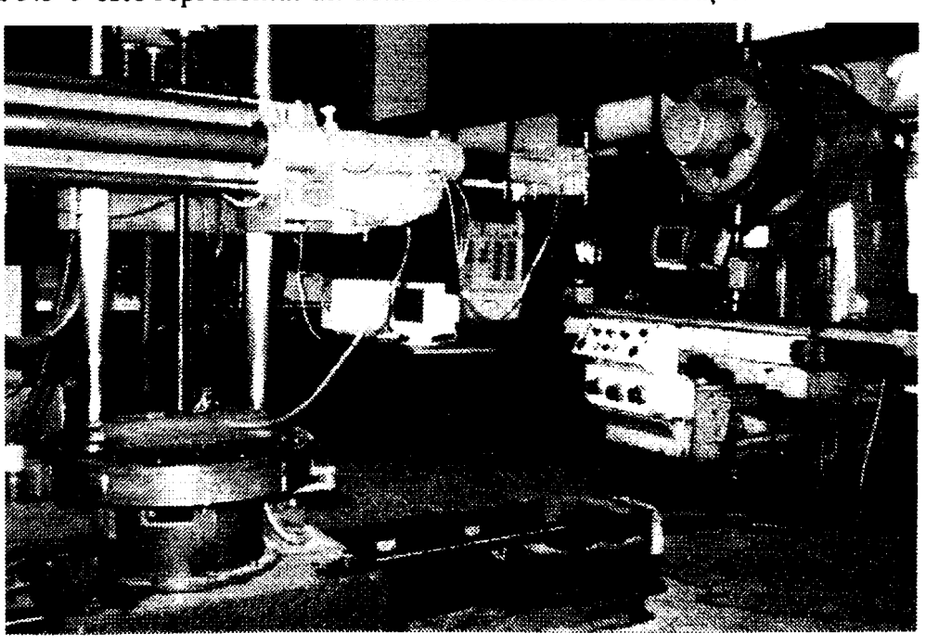


Figura 3.3-5

3.3.c.a. Celula de gestionare a conveiorului şi a depozitelor

Celula de gestionare a depozitelor şi a conveiorului conţine în primul rând un robot hidraulic "R1" pe post de manipulator. Robotul are structura RTT. Tot în acest post sunt incluse şi depozitele "D1" şi "D2" de tip matricial care adună piesele finite ("D1"), respectiv piesele rebut ("D2"). Conveiorul "C1" (vezi Figura 3.3-6) are rolul de dispozitiv de intrare în sistem, el conţinând semifabricatele care se prelucrează în sistem, într-o ordine aleatoare. Piesele sunt preluate de pe conveior de robotul "R1" şi depuse pe dispozitivul de transfer (AGV) aflat în poziţia 1 ("POZ-1" în figură). Piesele care ies din sistem sunt preluate din aceeaşi poziţie şi manipulate spre depozitele 1 sau 2 după necesităţi. Manipularea pieselor se face cu ajutorul braţului de robot care este echipat cu un dispozitiv de prehensiune cu trei degete comandat hidraulic. Piesele sunt prinse pe zona cilindrică de diametru ∅ 100 (vezi Figura 3.3-1 şi Figura 3.3-2).

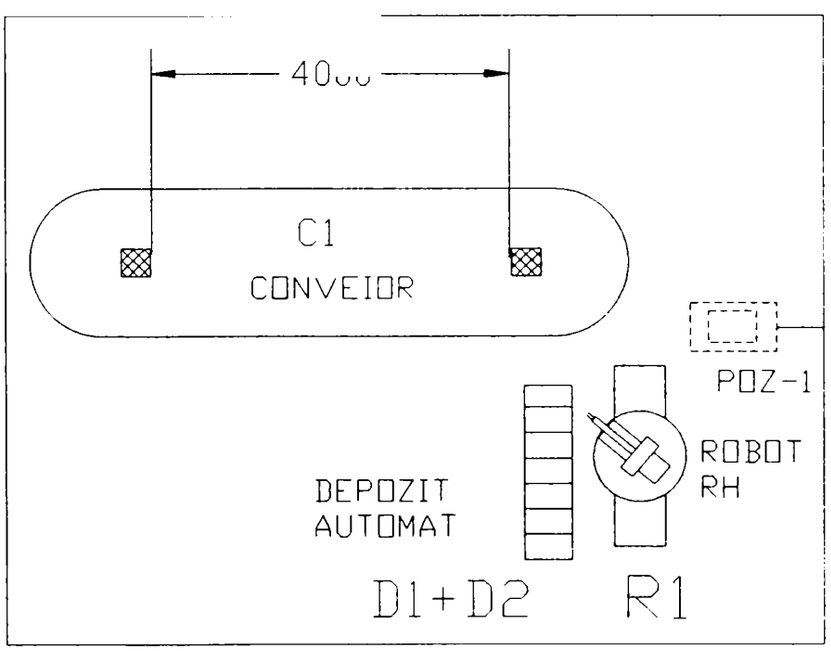


Figura 3.3-6

3.3.c.b. Celula de frezare

Se observă în Figura 3.3-7 că maşina de frezat este echipată cu un dispozitiv de lucru "DL1" cu acţionare pneumatică, dispozitiv care este folosit pentru fixarea pieselor pe masa maşinii de frezat la nivelul cotei ∅ 100 sau al cotei ∅ 150, după necesităţi. Dispozitivul are o cursă suficientă a bacurilor de fixare pentru a acoperi această diferenţă de diametre.

Servirea maşinii de frezat este asigurată de un robot "R2" de tip REMT-2, echipat cu o translaţie inferioară lung (2000 mm), care poate efectua astfel şi mişcări de transfer. Acesta

efectuează operațiile de alimentare a maşinii de frezat cu piesele care necesită astfel de operații, în speță piese de tip "Flanşa1". Piesele sunt preluate de pe dispozitivul de transfer AGV parcat în poziția 2 ("POZ-2" în figură) care le aduce în această poziție de la robotul "R1" de încărcare/descărcare. După prelucrare piesele se transferă cu ajutorul robotului "R2" pe masa robotului de măsurare.

Pe lângă sarcina de servire a maşinii de frezat, robotul "R2" mai are şi rolul de a manipula piesele de la postul de măsurare. Astfel, robotul va prelua din acest post piesele care ies de la măsurare cu verdictul "OK" şi le va transfera pe dispozitivul de transfer AGV unde se va face asamblarea cu piesa de tip "Flanşa2". Piesele care vor avea la măsurare verdictul "Rebut" vor fi de asemenea transferate pe AGV pentru transferul spre depozitul "D2".

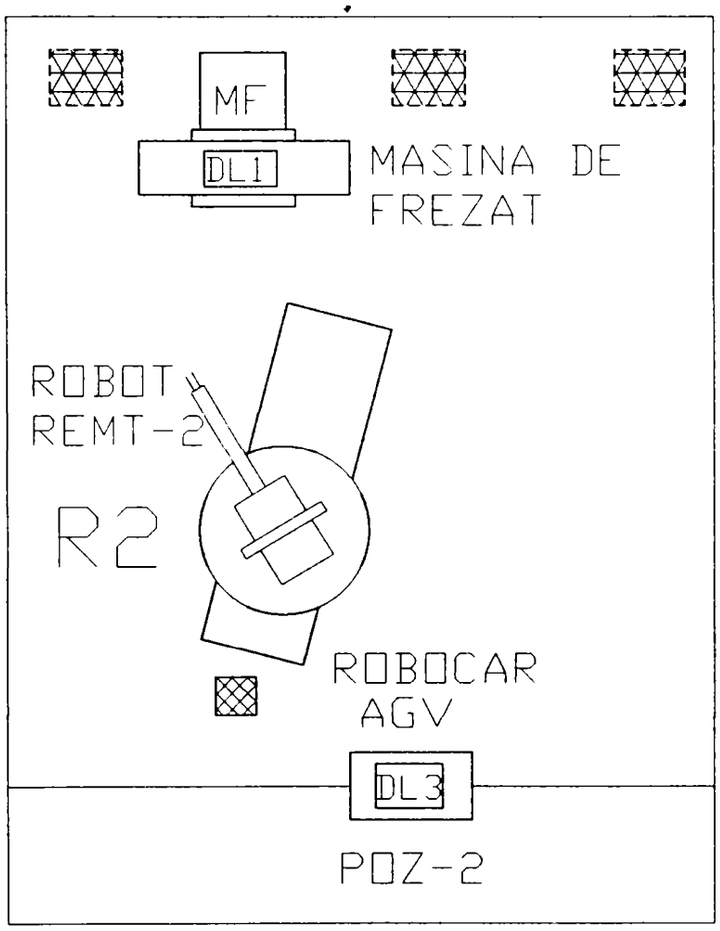


Figura 3.3-7

3.3.c.c. Celula de măsurare

Ciclul de măsurare presupune efectuarea operațiilor de verificare asupra reperelor care au fost prelucrate prin frezare, în vederea trecerii lor la asamblare sau, eventual, în magazia de rebuturi "D2". Măsurarea propriu-zisă se efectuează cu ajutorul unui palpator instalat în sistemul de prehensiune al robotului "R3" de tip REMT-5. Piesa se găseşte în

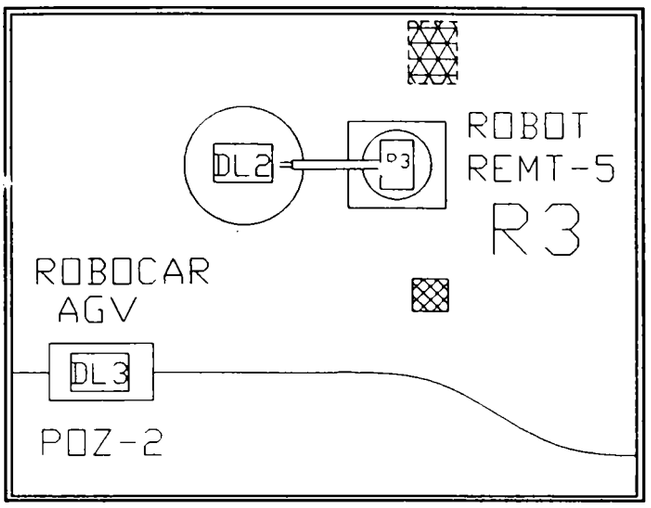


Figura 3.3-8

timpul măsurării aşezată într-un dispozitiv "DL2". Pentru efectuarea măsurătorilor sistemul are nevoie de setarea unui punct de referinţă pe suprafaţa piesei iar pentru calculul dimensiunilor sunt folosite cotele de poziţie ale robotului din momentul în care palpatorul atinge diverse puncte de pe suprafaţa piesei în conformitate cu planul de măsurători.

3.3.c.d. Celula de sudare

O altă celulă distinctă aflată în componenţa sistemului de fabricaţie flexibilă este celula de sudare, reprezentată în Figura 3.3-9. Această celulă este deservită de un robot de sudare "R4" de tip Cloos şi de o masă de poziţionare cu două posturi în care se găsesc două dispozitive de lucru "DL4".

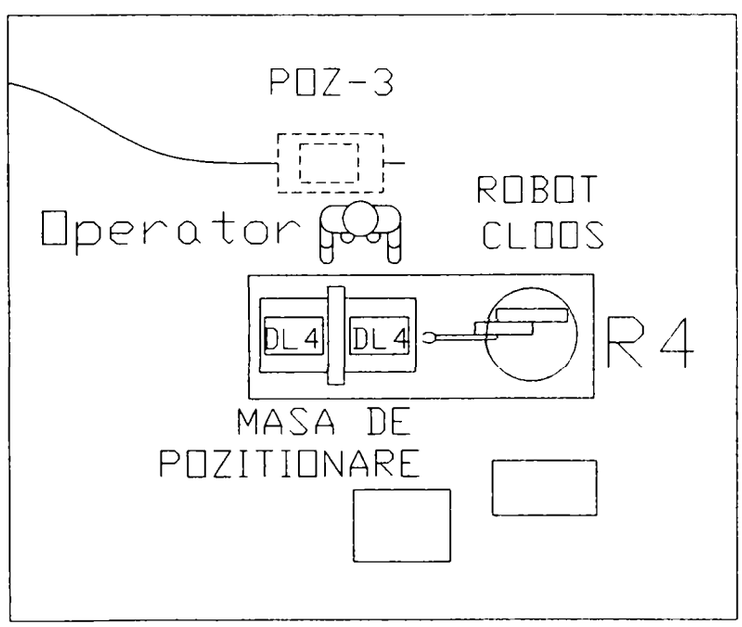


Figura 3.3-9

Ciclul de sudare beneficiază de serviciul unui operator uman. Acesta are rolul de a efectua încarcarea/descărcarea dispozitivelor de sudare şi transferul pieselor de pe dispozitivul de transfer AGV în zona robotului de sudare. Dispozitivul de transfer se află parcat în cea de-a treia poziţie fixă ("POZ-3" în figură). Robotul de sudare efectuează cordonul circular de sudură la îmbinarea celor două piese. In timpul sudării uneia din piese, în al doilea post al

mesei de sudare se realizează schimbarea piesei sudate cu un nou set de piese pregătit pentru sudare, astfel încât timpii de pregătire la sudare să fie cât mai reduşi.

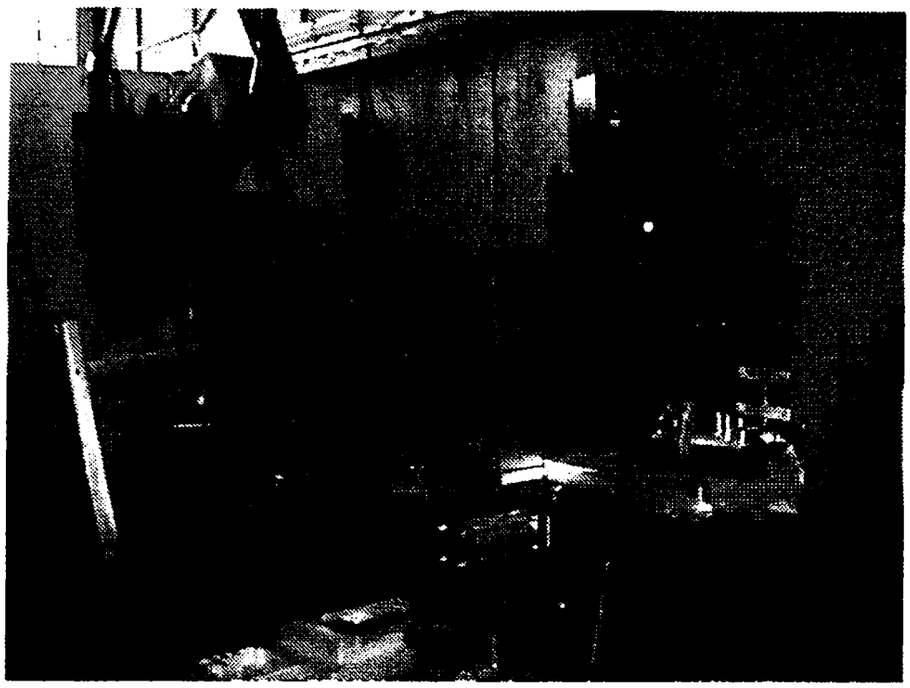


Figura 3.3-10

Sistemul prezintă particularitatea că unele dintre componentele sale sunt partajate de mai multe alte componente. In această situaţie se află dispozitivul de transfer de tip AGV, roboţii de servire a depozitelor şi a conveiorului, robotul de servire a maşinii de frezat. Această particularitate pune unele probleme de conflict în realizarea conducerii sistemului, probleme care au fost rezolvate prin alegerea unei logici de transfer corespunzătoare.

3.3.d Ciclogramele sistemului de fabricaţie

In sistem se introduc două tipuri de piese: piesa 1 ("Flanşa1") care are necesităţi de prelucrare şi piesa 2 ("Flanşa2") care nu se prelucrează ci doar se asamblează cu primul tip de piesă după ce aceasta a fost prelucrată. Asamblarea celor două piese se face la nivelul dispozitivului "DL3" aflat pe platforma superioară a dispozitivului de transfer. In urma asamblării în sistem va circula un nou tip de piesă denumită "Ansamblu". Această piesă va fi transferată la robotul de sudare "R4" de unde se va întoarce cu ajutorul AGV-ului în depozitul "D1".

Toate cele de mai sus sunt valabile în cazul în care după frezare rezultă o piesă "OK", adică o piesă bună dimensional. Dacă însă din procesul de frezare rezultă o piesă incorectă

dimensional, adică un "Rebut", atunci aceasta nu va mai fi transferată la asamblare ci va fi depozitată în depozitul "D2" unde ajunge prin manipularea cu "R2" şi transferul cu dispozitivul AGV, urmat de manipularea cu robotul "R1".

In ciclogramele care vor urma sunt prezentate operaţiile care se execută în sistem pentru prelucrarea pieselor de tip 1 (Figura 3.3-11) şi a pieselor de tip 2 (Figura 3.3-12).

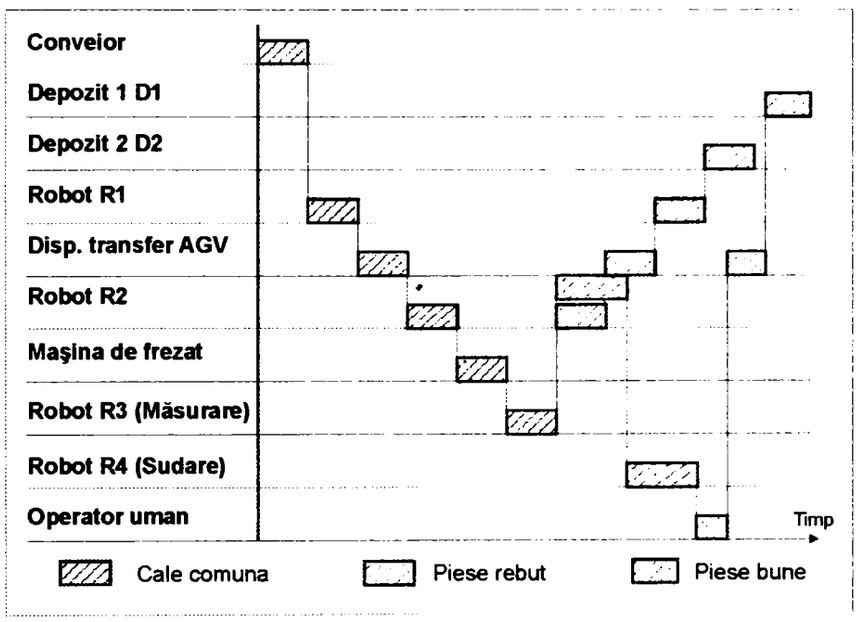


Figura 3.3-11

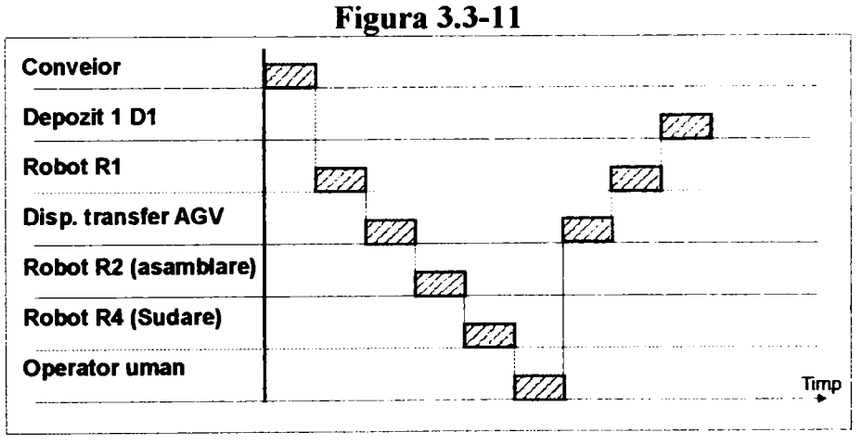


Figura 3.3-12

3.3.e Analiza structurală a sistemului de fabricaţie

Analiza structurală a sistemului a fost efectuată pe baza celor prezentate în capitolul I al lucrării. Sistemul are în componenţa lui un număr de 5 subsisteme interconectate. Legăturile dintre ele apar în Figura 3.3-13. Aşa după cum s-a afirmat şi în capitolul I, la ieşirea sistemului de fabricaţie se va regăsi un flux mixt, de materie şi informaţie, fapt care se explică prin faptul ca prin prelucrare mecanică se modifică în bună măsură conţinutul informaţional al reperelor în măsura în care forma lor se schimbă.

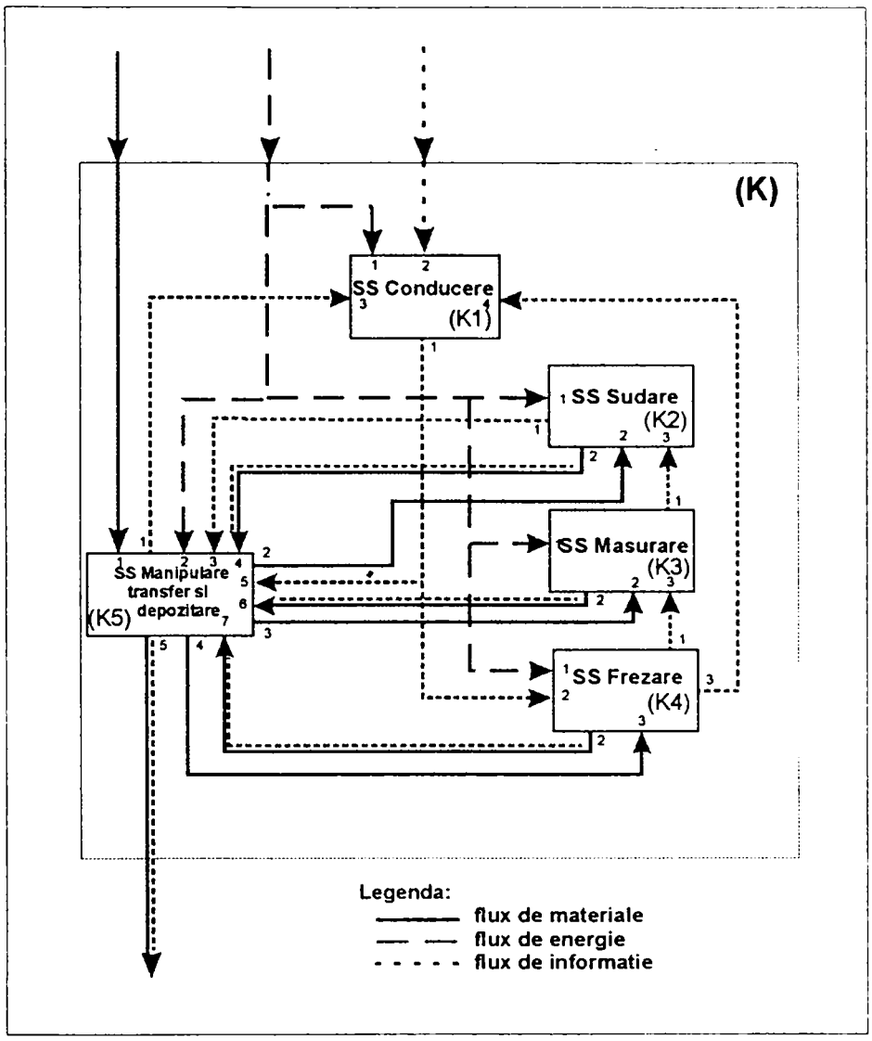


Figura 3.3-13

S-a considerat că dispozitivele de transport de tip AGV, conveiorul şi roboţii fac parte dintr-un singur subsistem şi anume subsistemul "K5". De asemenea, toate echipamentele de comandă au fost incluse într-un singur subsistem, "K1".

Matricea de structură a sistemului va avea dimensiunile 5X5 si va fi de forma:

$$K = \begin{bmatrix} 0 & 0 & 0 & K_{14} & K_{15} \\ 0 & 0 & 0 & 0 & K_{25} \\ 0 & K_{32} & 0 & 0 & K_{35} \\ K_{41} & 0 & K_{43} & 0 & K_{45} \\ K_{51} & K_{52} & K_{53} & K_{54} & 0 \end{bmatrix} \quad (3.3\text{-}1)$$

In continuare se vor prezenta câteva dintre *matricile de cuplare* ale sistemului.

$$K_{14} = [0 \quad 1 \quad 0] \qquad K_{15} = [0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0]$$

$$K_{25}=\begin{bmatrix}0&0&1&0&0&0&0\\0&0&0&1&0&0&0\end{bmatrix}\quad K_{32}=\begin{bmatrix}0&0&1\\0&0&0\end{bmatrix}$$

$$K_{45}=\begin{bmatrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&1\\0&0&0&0&0&0&0\end{bmatrix}$$

(3.3-2)

Unul dintre aspectele principale ale studiului desfăşurat în lucrarea de faţă se referă la arhitectura de conducere a sistemului de fabricaţie. Aşa după cum a fost descris mai devreme, sistemul de fabricaţie flexibilă nu are un soft de conducere centralizată, aşa încât doar celula de frezare poate fi controlată în mişcarea coţelată cu robotul de servire al maşinii de frezat "R2". Pentru această celulă a fost conceput în faza de proiectare a ei un program C pentru controlul celor 4 axe ale robotului "R2" precum şi pentru transferul secvenţial al frazelor programelor NC la maşina de frezat. Celelalte celule din componenţa sistemului funcţionează secvenţial, pe baza unor confirmări primite de la celulele cu care cooperează pe fluxul de prelucrare, fiind conduse de controlere locale. Pentru a putea evidenţia diferenţele care apar în arhitectura de conducere a sistemului în urma implementării unei noi metode de conducere propusă de autor, s-a elaborat şi o schemă structurală a sistemului actual de conducere care este prezentată în Figura 3.3-14.

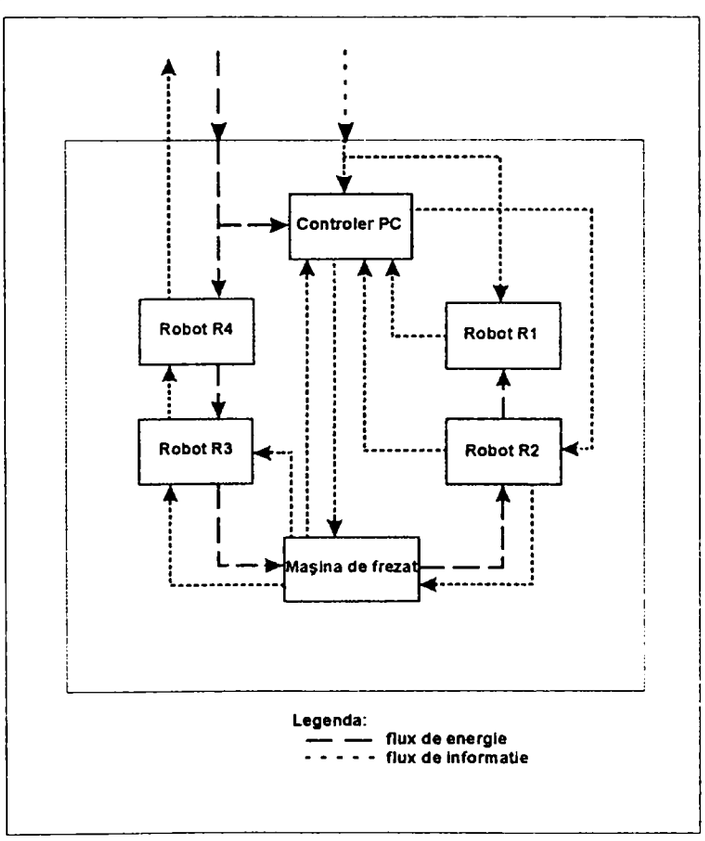


Figura 3.3-14

Este demn de remarcat faptul că pentru arhitectura actuală a sistemului de conducere controlerul PC are schimb de informaţie doar cu roboţii "R1" şi "R2" precum şi cu maşina de frezat. Celelalte echipamente sunt conectate informatic în mod secvenţial.

3.4. Modelul cu reţele Petri folosit în analiza parametrilor cantitativi ai sistemului de fabricaţie

Din acest motiv resursele partajate apar pe toate diagramele din model care conţin nodurile deservite.

In analiza performanţelor unui sistem de fabricaţie se folosesc multe procedee şi metode, dintre care se pot aminti metoda şirurilor de aşteptare, metoda lanţurilor Markov, metoda reţelelor Petri stocastice. Pentru "lay-out"-ul din Figura 3.3-4 s-a elaborat un model sub forma unei diagrame cu reţele Petri. Programul de calcul utilizat pentru realizarea diagramei şi rularea simulărilor este "DesignCPN" pe o platformă LinuxPC. In scopul evidenţierii importanţei strategiei de rutare (de transfer) în funcţionarea sistemului, diagramele din model au fost realizate în două variante:

- varianta I, la care s-a modelat o logică de transfer a materialelor foarte simplă, bazată în principal pe *partajarea resurselor* şi pe câteva reguli de excludere a unor marcaje care duc în mod evident la blocaje;
- varianta II, la care modelul include o logică de transfer mult mai severă, bazată tot pe partajarea resurselor dar în prezenţa unor *restricţii impuse de tehnologia de execuţie* a pieselor. Acest model conţine o pagină suplimentară faţă de varianta I, în care este implementată această nouă logică de rutare, pagina "general".

In Anexa 3-1 şi Anexa 3-2 sunt prezentate toate paginile celor două diagrame.

Semnificaţia claselor (color) şi a variabilelor (var) folosite în model este prezentată în paginile "declaraţii" şi "temporar" din cele două modele, aşa cum apar ele în anexe. După cum se poate observa, a fost declarată mai întâi o clasă de tip "Piesa". Aceasta conţine variabile şir care sunt definite de utilizator în momentul precizării tipurilor de piese de pe conveior. Aceste variabile au conţinut identic cu numele pieselor ce se vor încărca în sistem. O altă clasă este "Paleta". Ea conţine un produs de cinci variabile, dintre care primul este numele piesei, iar celelalte sunt destinaţiile pieselor aflate pe paletă, după cum urmează: 0-nici o destinaţie, 1-frezare, 2-măsurare, 3-asamblare, 4-sudare. Piesele denumite "Flansa1" sunt piese care nu necesită prelucrări în sistem. Ele se asamblează cu piesele "Flansa2" şi generează prin asamblare piesele "Ansamblu". Structura reperului "Ansamblu" este

1"Flansa1" + 1"Flansa2". Sistemul poate genera şi piese de tip "Rebut", într-un procent controlat prin funcţia "Ok(s)". Valorile parametrului "s" sunt generate aleator de către sistem cu o distribuţie uniformă pe intervalul de numere întregi 0...100. Valoarea generată este apoi comparată cu un prag care dă limita de rebuturi. Aceste piese rezultă în urma prelucrărilor de frezare a reperelor de tip "Flansa1".

Itinerarul tehnologic care a stat la baza modelului elaborat este redat în Figura 3.4-1

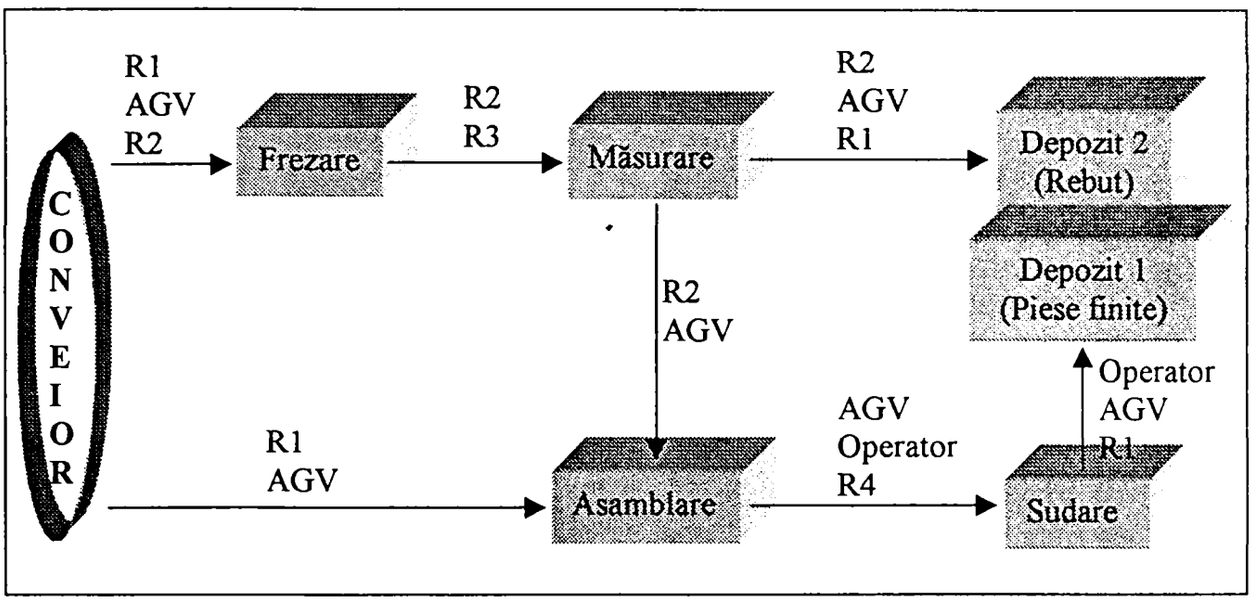


Figura 3.4-1

Metoda de simulare bazată pe utilizarea reţelelor Petri permite considerarea parametrului 'timp". Acesta apare în inscripţiile de pe arcele de legătură dintre poziţii şi tranziţii sau la nivelul tranziţiilor sub forma unor "time region". In acest mod toate marcajele din sistem care sunt modificate prin declanşarea tranziţiilor cu inscripţii de timp vor avea o "etichetă" de timp care se modifică în timpul rulării modelului. Pe toate paginile din diagrame pot fi observate în anexe inscripţiile de timp care apar sub forma "@+<întârziere>".

Una din problemele majore ale funcţionării în ciclu automat a sistemelor de fabricaţie o constituie apariţia blocajelor. Acestea pot fi de mai multe feluri, în funcţie de cauzele care le produc. Pe baza analizei făcute în capitolul 1 asupra acestui fenomen de apariţie a blocajelor se poate afirma că în cazul modelului elaborat în prezenta lucrare este vorba de *blocaje de excludere mutuală*, care apar datorită faptului că resursa "AGV" este partajată de mai multe noduri ale reţelei şi atunci când ea este cerută de un proces poate fi chiar atunci folosită de un altul, apărând astfel un blocaj şi sistemul se va opri. In Figura 3.4-2 este redată starea de blocaj a sistemului datorată resursei "AGV".

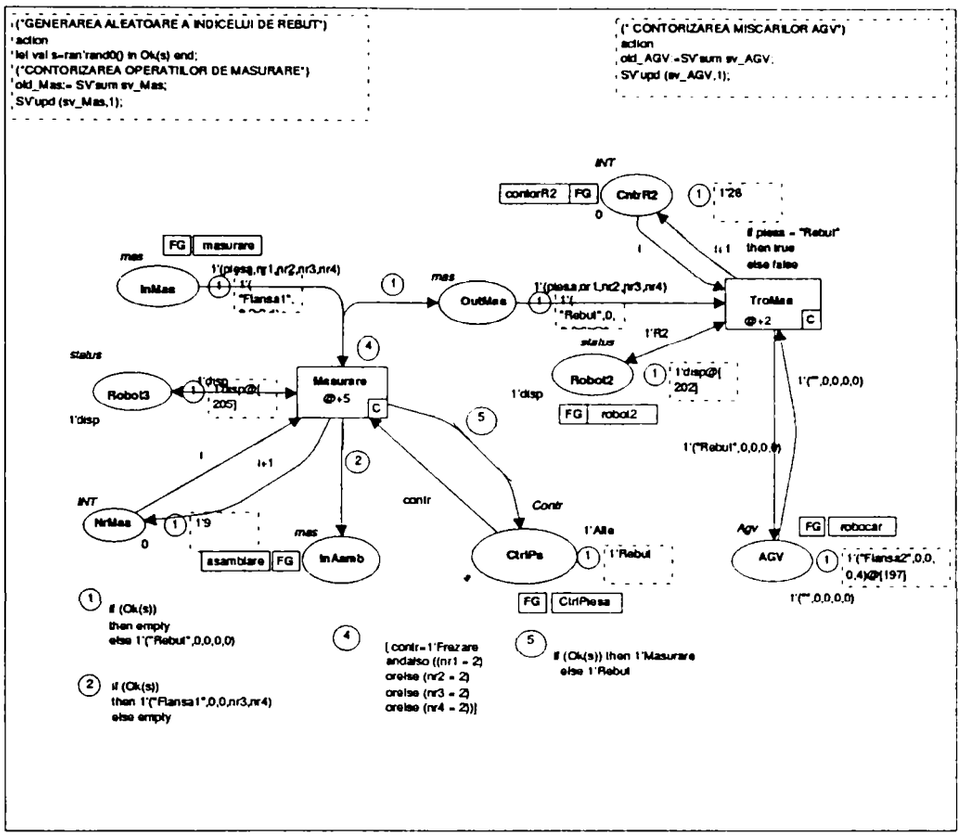


Figura 3.4-2

Blocajul apare datorită generării aleatoare a rebuturilor. Sistemul funcționând doar pe baza logicii simple de transfer bazată pe partajarea resurselor, în momentul în care dispozitivul de transfer "AGV" este liber el poate accepta o piesă de tip "Flanşa2", aleasă la întâmplare de pe conveior, piesă care va aştepta în dispozitiv terminarea frezării piesei de tip "Flanşa1". Dacă operația de frezare va genera un rebut, cum dispozitivul de transfer este deja ocupat cu piesa de tip "Flanşa2" care nu necesită prelucrări, acesta (rebutul) nu mai poate fi transferat la depozitul de rebuturi, deci întregul sistem se va bloca şi trebuie oprit. In cazul celei de-a doua variante de logică de transfer, bazată pe tehnologia de execuție a reperelor, blocajele de excludere mutuală sunt complet evitate, datorită eliminării din şirul marcajelor posibile ale acelora care duc la blocaje.

3.5. Elemente de analiză cantitativă

Programul de modelare/simulare utilizat oferă posibilitatea de analiză cantitativă a funcționării sistemului. Prima dintre posibilități este de *a urmări modificările cantitative* ce se produc, *direct pe diagramele din model*. Pe diagramele din anexe se poate observa că fiecare poziție are în vecinătatea ei un chenar ("current marking") în care este redat marcajul curent şi etichetele de timp ale elementelor ("token") care îl compun.

O altă modalitate de analiză cantitativă a unui sistem pe baza modelării o constituie *generarea dinamică a unor diagrame*, cu ajutorul programului de modelare. Fiecare din cele

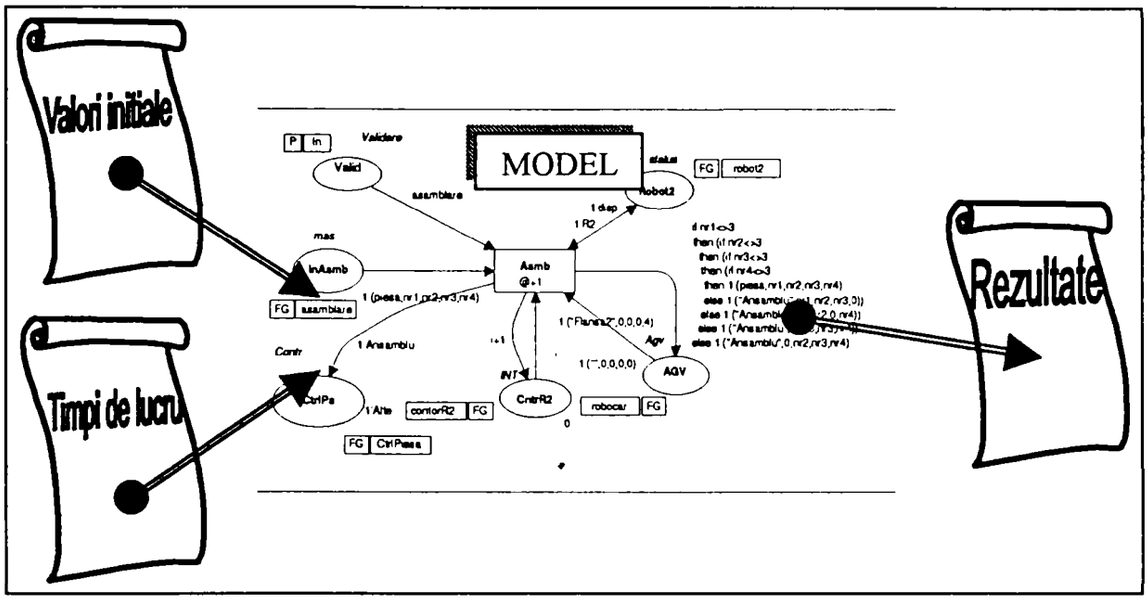


Figura 3.5-1

două variante de diagrame din anexe conține câte o pagină denumită "diagrame", unde apar astfel de reprezentări grafice ale elementelor cantitative din model. Prelevarea valorilor parametrilor reprezentați în diagrame pe parcursul desfăşurării simulării se face cu ajutorul unor variabile de tip special, numite *variabile statistice* şi care sunt definite inițial în pagina de declarații din model.

Cea mai flexibilă metodă de determinare a parametrilor cantitativi ai modelului, metodă folosită şi în lucrarea de față, constă în utilizarea unor secvențe de cod asociate unora dintre tranzițiile sistemului, secvențe care actualizează valorile variabilelor statistice pe de o parte, sau calculează anumiți parametrii care apoi sunt salvați în fişiere speciale ce se vor prelucra ulterior cu metode numerice specifice. In Figura 3.5-1 sunt redate componentele procesului de simulare, în care sunt evidențiate, pe lângă modelul propriu-zis şi fişierele de intrare care conțin parametrii inițiali, precum şi cele de ieşire, unde sunt acumulate rezultatele simulării. O primă pagină dintr-un fişier de date se poate consulta în Anexa 1 a prezentului capitol. Toate mărimile care apar în acest fişier sunt determinate direct din simulare, fie ca valori momentane ale variabilelor statistice, fie ca valori calculate cu ajutorul acestora. Modul în care se constituie fişierul de date poate fi consultat în paginile "temporar" din model, cuprinse de asemenea în anexe, în secțiunea ocupată de funcția "ScrieValStat()". Prezenta analiză cantitativă are drept scop determinarea influenței logicii de transfer (rutare) asupra parametrilor cantitativ ai sistemului. Pentru cele două variante de modele cu logici de transfer

diferite după cum s-a arătat mai sus, s-au extras din fişierele de date obţinute din simulare următoarele valori:

- timpul principal de fabricaţie (TPF), considerat de la începutul simulării până la oprirea ei, indiferent de cauzele care au determinat-o;
- numărul de mişcări ale dispozitivului de transfer (AGV);
- numărul de piese finite (Fin);
- numărul de piese frezate (Fre);
- numărul de piese sudate (Sud);
- numărul de rebuturi (Reb);
- numărul de piese în ciclu (Pic1 şi Pic2);
- timpul mediu de transfer al unei piese (t_a);
- timpul mediu de lucru la frezare (t_f);
- timpul mediu de lucru la sudare (t_s);
- numărul de piese de tip 1 rămase neprelucrate (NF1);
- numărul de piese de tip 2 rămase neprelucrate (NF2);
- numărul total de piese de tip 1 din lot (TF1);
- numărul total de piese de tip 2 din lot (TF2).

Pe baza acestor valori s-au determinat prin calcul următorii *parametrii cantitativi ai funcţionării sistemului*:

- timpul principal de fabricaţie (TPF), definit de relaţia

$$TPF = \frac{TTL}{Fin} \ [\text{min/piesa}]; \qquad (3.5\text{-}1)$$

- indicele de utilizare al dispozitivului de transfer (u_{agv}), definit de relaţia

$$u_{agv} = \frac{AGV * t_a}{TTL} \times 100 \quad [\%]; \qquad (3.5\text{-}2)$$

- indicele de utilizare al maşinii de frezat (u_f), definit de relaţia

$$u_f = \frac{Fre * t_f}{TTL} \times 100 \quad [\%] \qquad (3.5\text{-}3)$$

- indicele de utilizare al robotului de sudare (u_s), definit de relaţia

$$u_s = \frac{Sud * t_s}{TTL} \times 100 \quad [\%]$$

(3.5-4)

- indicele de blocaj (i_b), definit de relația

$$i_b = \frac{NF1 + NF2}{TF1 + TF2} \times 100 \quad [\%]$$

(3.5-5)

- indicele de blocaj a pieselor în ciclu (i_{bc}), definit de relația

$$i_{bc} = \frac{Pic1 + Pic2}{TF1 + TF2} \times 100 \quad [\%]$$

(3.5-6)

Prin intermediul acestor parametrii poate fi caracterizată în mod global comportarea unui sistem de fabricație. Evoluția lor spre optim este însă contradictorie, ținând cont că unii dintre ei se pot influența reciproc. Considerați însă individual, parametrii prezentați mai sus au valorile de optim conform datelor din Tabelul 3.5-1.

Tabelul 3.5-1

Parametrul	**TPF**	u_{agv}	u_f	u_s	i_b	i_{bc}
	[min/piesă]	[%]	[%]	[%]	[%]	[%]
Tendința optimă	↘	↗	↗	↗	↘	↘
Valoarea optimă	Suma timpilor de maşină	100	100	100	0	0

Parametrii globali ai simulărilor au fost:

- *mărimea lotului de fabricație*, exprimată prin numărul de piese de tip 1 şi 2 care au fost aşezate în conveiorul de intrare în sistem;
- *procentajul de rebuturi admis*, care este de 15%, respectiv 10%.

Valorile relativ mari ale procentajului de rebuturi se justifică prin necesitatea de a putea evidenția influențele acestui parametru. Ele nu sunt neapărat valori realiste.

Toate determinările de coeficienți şi indici de performanță s-au făcut pentru diverse combinații ale acestor parametrii. Valorile parametrilor globali ai încercărilor au fost preluate din fişiere text existente pe sistemul de calcul. In acest fel, pentru acelaşi model, la fiecare rulare a simulării au putut fi utilizate diverse valori ale parametrilor globali ai încercărilor.

Tabelul 3.5-2

Indici	Rebut [%]		30_25		60_50		120_100	
			partajarea resurselor	transfer tehnologic	partajarea resurselor	transfer tehnologic	partajarea resurselor	transfer tehnologic
TPF	15 %	M	26,135	36,0836	30,4445	36,1796	30,4445	36,1796
		D	11,9441	1,91369	10,7855	1,18431	10,7855	1,18431
	10 %	M	25,9225	33,9133	24,8828	33,88	24,8828	33,88
		D	6,42080	0,88415	8,80681	0,75299	8,80681	0,75299
U_{agv}	15 %	M	25,4661	24,2574	20,6769	24,2284	20,6769	24,2284
		D	8,36251	0,27128	2,66314	0,19345	2,66314	0,19345
	10 %	M	31,6834	25,2250	20,7683	25,2318	20,7683	25,2318
		D	7,42033	0,15810	3,48246	0,13356	3,48246	0,1335
U_{fre}	15 %	M	35,2281	19,9138	32,6398	19,8264	32,6398	19,8264
		D	8,57624	0,29897	9,12893	0,22431	9,12893	0,22431
	10 %	M	30,9440	20,4591	34,6860	20,2951	34,6860	20,2951
		D	5,31636	0,20055	10,6271	0,17291	10,6271	0,17291
U_{sud}	15 %	M	24,6817	13,8907	21,1160	13,8340	21,1160	13,8340
		D	9,85255	0,69331	9,47933	0,47192	9,47937	0,47192
	10 %	M	21,0211	14,7533	24,6602	14,7650	24,6602	14,7650
		D	5,85254	0,39503	10,9701	0,32888	10,9701	0,32888
I_b	15 %	M	64,2045	2,20779	89,6212	1,21212	89,6212	1,21212
		D	38,3558	2,03948	9,14575	1,83190	9,14575	1,83190
	10 %	M	40,2424		80,6060		80,6060	
		D	37,9887		20,8986		20,8986	
I_{bc}	15 %	M	6,0227	1,03896	2,91666	0,45454	2,91666	0,45454
		D	2,45903	0,93373	1,00230	0,47475	1,00230	0,47475
	10 %	M	5,93939	1,74242	3,28282	0,84639	3,28282	0,84639
		D	2,22327	0,65207	1,21323	0,23443	1,21323	0,23443

In Tabelul 3.5-2 sunt concentrate toate valorile medii obținute în urma simulărilor, pentru loturi de fabricație de 30+25, 60+50, respectiv 120+100 de piese de tip "Flanşa1" şi

"Flanşa2". Variaţia tuturor indicilor de performanţă ai sistemului poate fi consultată în graficele cuprinse în anexele prezentului capitol.

Analizând tabelul menţionat mai sus se pot formula mai multe *observaţii generale*:

- indicii de performanţă nu depind practic de mărimea lotului de fabricaţie pentru aceeaşi variantă de logică de transfer;
- logica de transfer influenţează decisiv valoarea indicilor de performanţă;
- procentajul de rebuturi influenţează evident parametrul TPF;
- dispersiile indicilor de performanţă calculaţi (D) sunt mult mai mari în cazul logicii de transfer cu partajarea resurselor (varianta 1) decât în cazul logicii bazate pe tehnologia de execuţie (varianta 2)

O analiză mai detaliată a rezultatelor din Tabelul 3.5-2 va fi desfăşurată în cele ce urmează.

3.5.a Timpul principal de fabricaţie

In graficele de mai jos se poate observa că parametrul TPF este practic constant în funcţie de mărimea lotului de fabricaţie. Acest lucru se explică prin faptul că ponderea timpului de maşină în totalul timpului de lucru este aceeaşi indiferent de numărul de piese din lot. In ambele variante de logică de transfer nu apar influenţe semnificative ale mărimii lotului asupra parametrului TPF

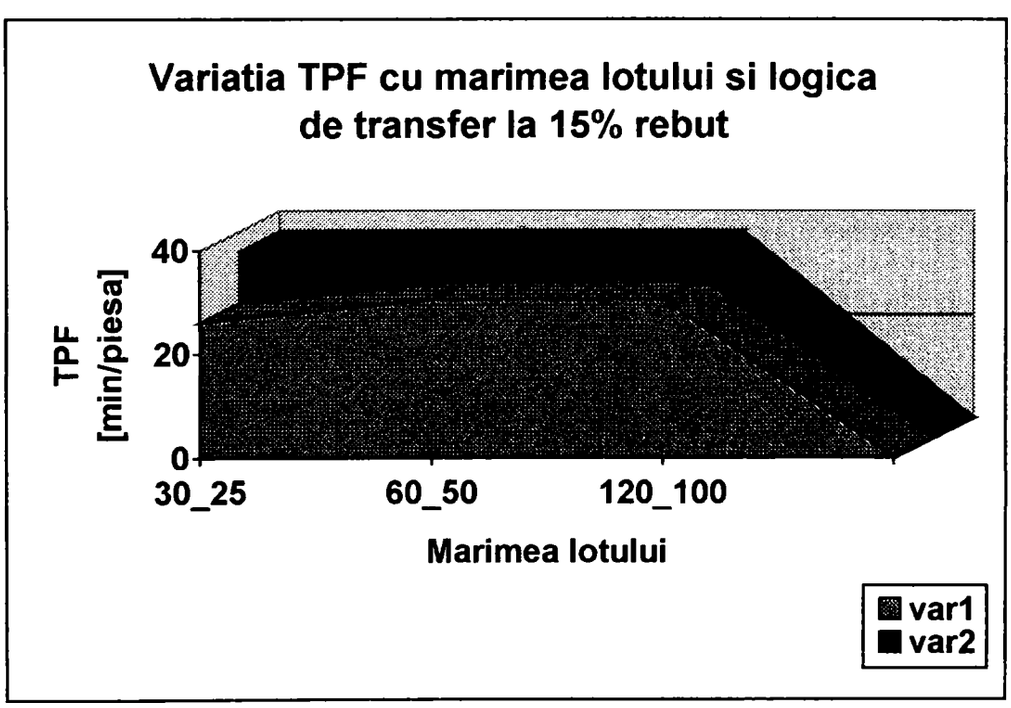


Figura 3.5-2

Dacă ne referim însă la logica de transfer, se poate observa că în varianta 1 valorile parametrului analizat sunt mai mici decât în varianta 2 pentru toate loturile simulate. Această diferență se poate explica prin faptul că opririle datorate blocajelor nu mai permit efectuarea unor operații de transfer sau de alimentare care "încarcă" bilanțul de timp al sistemului. O altă explicație ar fi legată de faptul că varianta 2 de logică de transfer, bazată pe tehnologia de execuție este mult mai rigidă decât logica bazată pe simpla partajare a resurselor şi nu permite suprapunerea unor operații cu pericol de determinare a blocajelor. In acest fel timpul sistemului pentru acelaşi număr de piese prelucrate va creşte, deci va creşte implicit şi valoarea parametrului TPF. In graficul următor este redată variația aceluiaşi parametru pentru nivelul de rebuturi de 10%. Datorită scăderii procentajului admis de rebuturi se obține o variație mult mai mică a valorilor parametrului TPF cu mărimea lotului de fabricație în comparație cu primul caz.

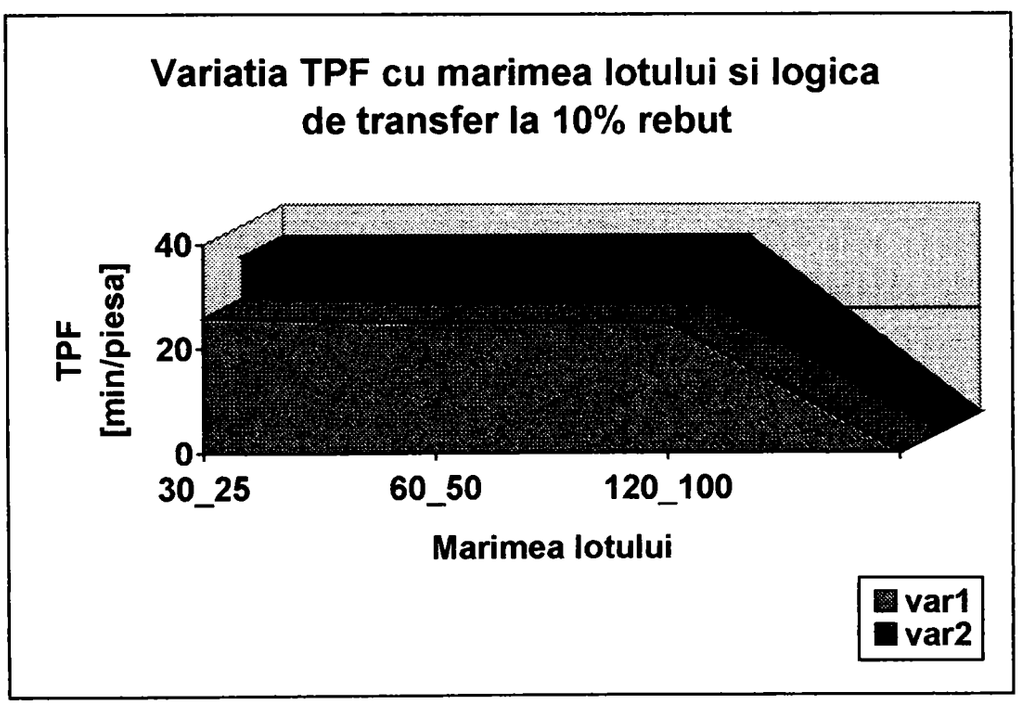


Figura 3.5-3

3.5.b Indicele de utilizare al dispozitivului de transfer U_{agv}

Numărul de mişcări efectuat de dispozitivul de transfer de tip AGV este apreciat prin intermediul indicelui de utilizare al dispozitivului de transfer U_{agv} . La fel ca şi în cazul anterior, acest indicator de performanță al sistemului are valori mai mari în cazul logicii de transfer varianta 2. Acest lucru denotă faptul că în cazul unei logici bazate exclusiv pe

partajarea resurselor este mai puțin folosit. Graficul următor redă spre exemplificare variația indicelui analizat cu mărimea lotului şi logica de transfer pentru cazul 10% rebut.

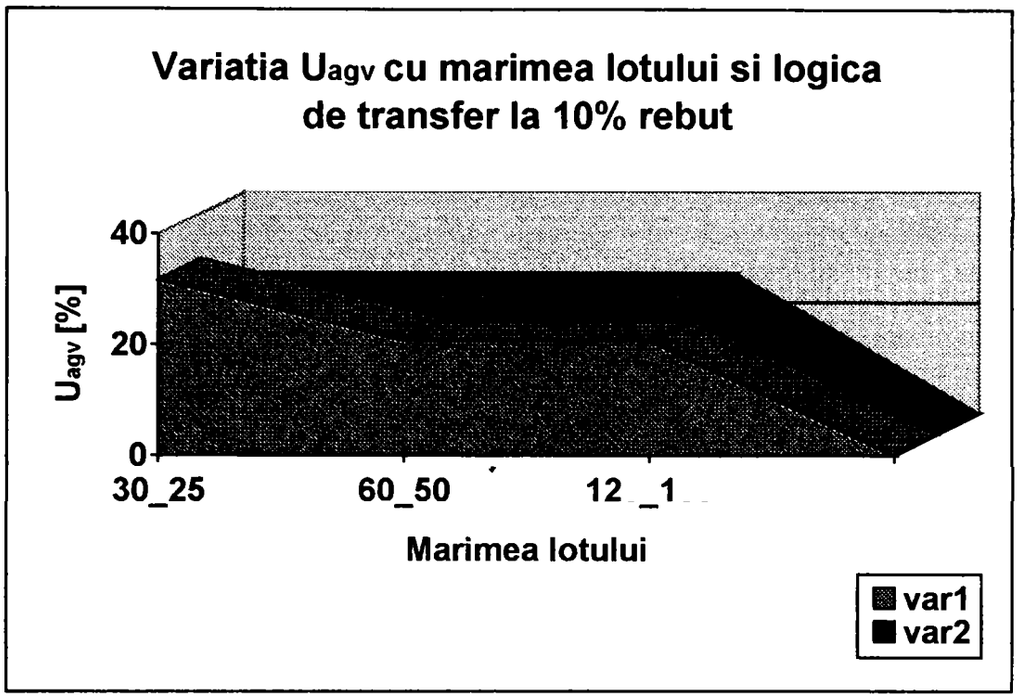


Figura 3.5-4

3.5.c Indicele de utilizare al maşinii de frezat (U_{fre}) şi indicele de utilizare al robotului de sudare(U_{sud})

Aceşti doi indici vor fi analizați împreună, deoarece conform datelor din Tabelul 3.5-2 variația lor în timpul simulării este aproape identică. Se poate observa faptul că valorile indicilor scad în cazul variantei 2 a logicii de transfer. Valorile mai reduse ale celor doi indici vin să confirme realitatea care a demonstrat deja o utilizare mai intensă a dispozitivului de transfer atunci când logica de transfer se bazează pe itinerarul tehnologic al piesei, ceea ce înseamnă că din bilanțul general de timp al modelului se foloseşte pentru freză şi robotul de sudare un timp ceva mai redus. Diagramele din Figura 3.5-5 şi Figura 3.5-6 redau variația indicilor "U_{fre}" şi "U_{sud}" pentru cazul 10% rebut în cele două variante ale logicii de transfer.

O altă observație bazată pe rezultatele simulărilor repetate este că valorile indicilor se mențin practic constante în funcție de procentajul de rebuturi admis. Acest lucru este evident mai ales în cazul logicii de transfer tehnologic, ceea ce denotă că sistemul echipat cu o astfel de logică este mult mai stabil. Simpla partajare a resurselor nu este întotdeauna soluția cea mai favorabilă pentru logica de rutare, din mai multe motive, unul dintre ele fiind şi faptul că nu se poate asigura o repetabilitate acceptabilă a funcționării sistemului din cauza blocajelor care apar aleator în sistem.

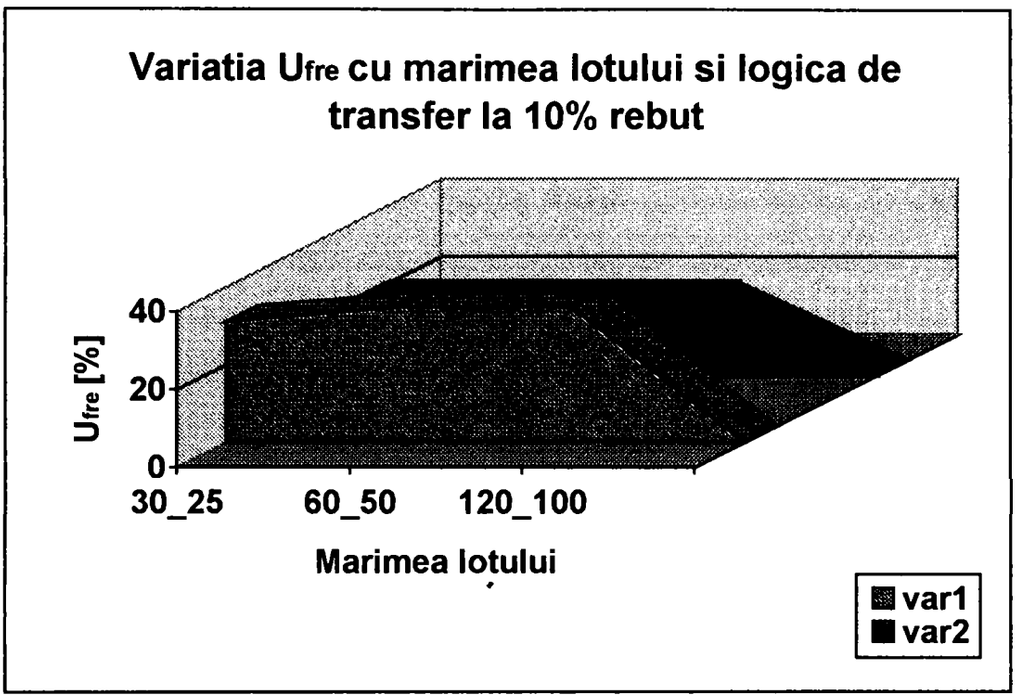


Figura 3.5-5

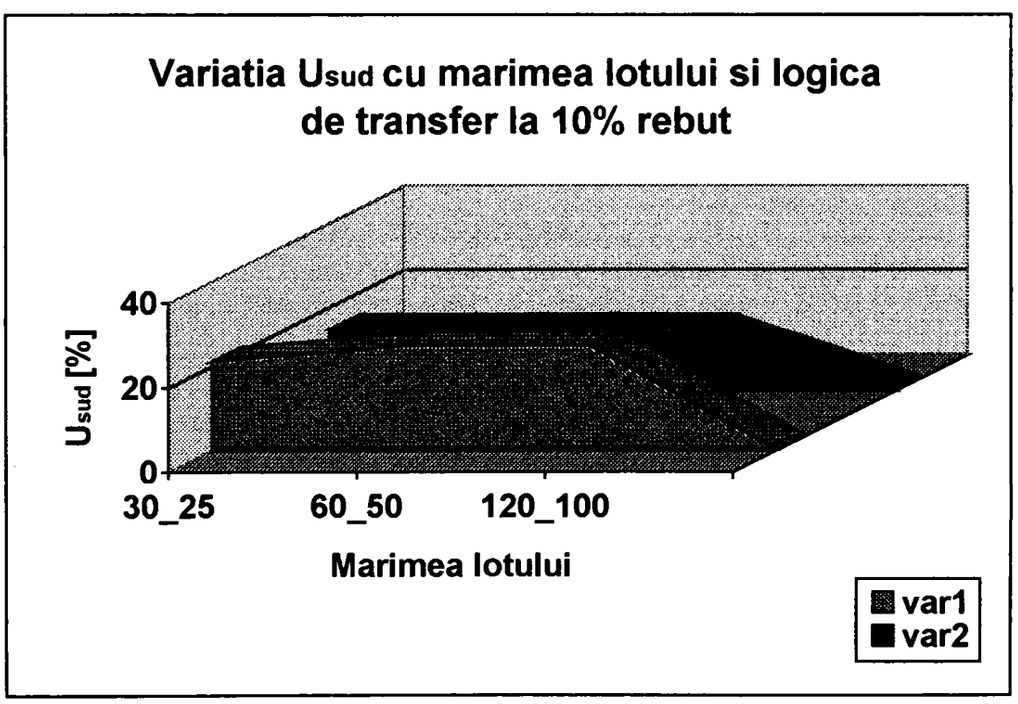


Figura 3.5-6

3.5.d Indicele de blocaj I_b

Parametrul care depinde în cel mai ridicat grad de tipul logicii de transfer este indicele de blocaj. Acesta a fost definit în relația (3.5-5) sub forma unui raport între numărul de piese din lot neprelucrate în momentul apariției blocajului şi numărul total de piese din lot. Definit astfel, coeficientul de blocaj poate fi interpretat ca un coeficient de nesiguranță al sistemului,

exprimând riscul ca sistemul să intre într-o formă de blocaj înainte de epuizarea întregului lot de piese de prelucrat. Se observă astfel că în cazul organizării transferului pe baza partajării resurselor, riscul de blocaj este foarte ridicat, variind între 64,2% şi 89,6%. Pentru cazul transferului tehnologic, riscul de blocaj scade foarte mult, la valori de câteva procente, datorate mai ales faptului că lotul de piese deşi echilibrat iniţial, adică cu o diferenţă între numerele de piese de cele două tipuri egală cu numărul de piese posibil de rebutat, devine dezechilibrat prin generarea aleatoare de rebuturi şi astfel vor rămâne piese neprelucrate de un tip sau altul.

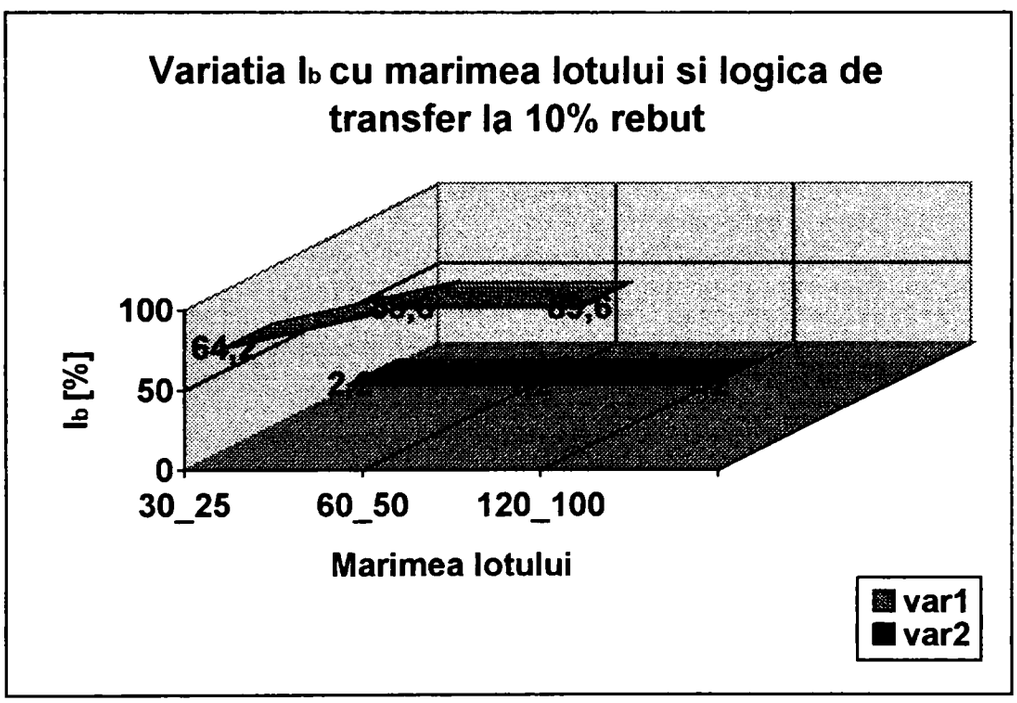


Figura 3.5-7

Variaţia indicelui de blocaj este redată în figura de mai sus. Valorile foarte mici ale acestuia pentru varianta 2 a logicii de transfer sunt o dovadă a utilităţii organizării transferului pe principii tehnologice.

Din analiza valorilor din Tabelul 3.5-2 se poate remarca şi lipsa indicilor de blocaj în cazul procentajului de rebut de 10%. In această situaţie nu au mai rămas piese neprelucrate şi astfel indicele de blocaj a avut valori nule. Practic întregul lot de piese a fost prelucrat, în condiţiile în care lotul a fost echilibrat în sensul prezentat mai sus. De asemenea se poate remarca faptul că valorile indicilor de blocaj sunt în general mai mici pe măsură ce scade procentajul de rebuturi admis la frezare. In mod ideal, dacă rebuturile ar apare cu probabilitate nulă, indicele de blocaj ar fi şi el nul.

3.5.e Indicele de blocaj a pieselor în ciclu I_{bc}

Acest parametru "măsoară" cantitatea de piese care se găsesc în ciclu în momentul apariţiei unui blocaj sau la epuizarea lotului. Aceste piese sunt de cele mai multe ori greu de recuperat sau de reintegrat în ciclu, fapt pentru care numărul de piese în ciclu trebuie să fie redus la un minim impus de funcţionarea eficientă a sistemului. Procentajul calculat este în raport cu numărul total de piese din lot. Si în acest caz, la fel ca şi în cazurile anterioare, valorile sunt mai ridicate în cazul logicii de transfer varianta 1, care presupune un grad mai mare de incertitudine în funcţionarea sistemului. In a doua variantă a logicii de transfer, când practic blocajele sunt eliminate, numărul de piese în ciclu provine din echilibrarea incorectă a lotului şi din jocul imprevizibil al apariţiei rebuturilor. In Figura 3.5-8 este redată variaţia parametrului I_{bc} în cazul procentajului de 15% rebut.

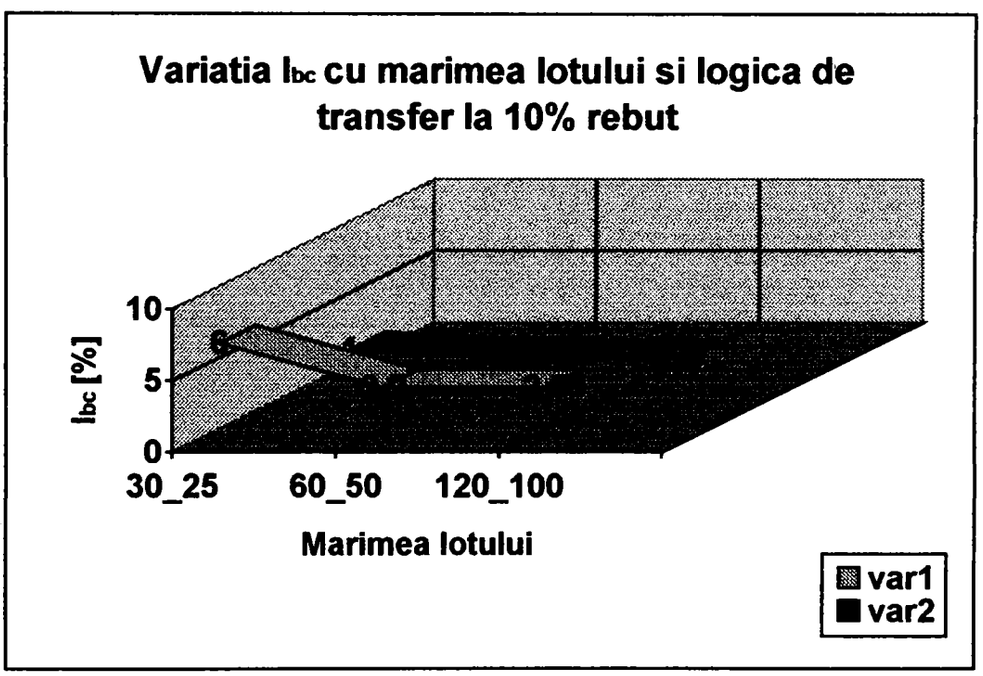


Figura 3.5-8

Fireşte că pe lângă generarea aleatoare de rebuturi în sistem mai pot apare şi alte surse de întreruperi aleatoare care nu au fost luate în considerare în această etapă. Modelul permite de pildă generarea unor valori aleatoare pentru timpii de maşină, care se citesc dintr-un fişier extern la fiecare validare a tranziţiilor corespunzătoare (vezi diagramele din anexe). Se pot de asemenea declara indisponibile anumite resurse ale sistemului, cum ar fi roboţii de servire ai maşinilor sau dispozitivului de transfer. Toate aceste procedee, care apropie din ce în ce mai mult modelul de lumea reală, nu fac decât să sporească gradul de incertitudine în

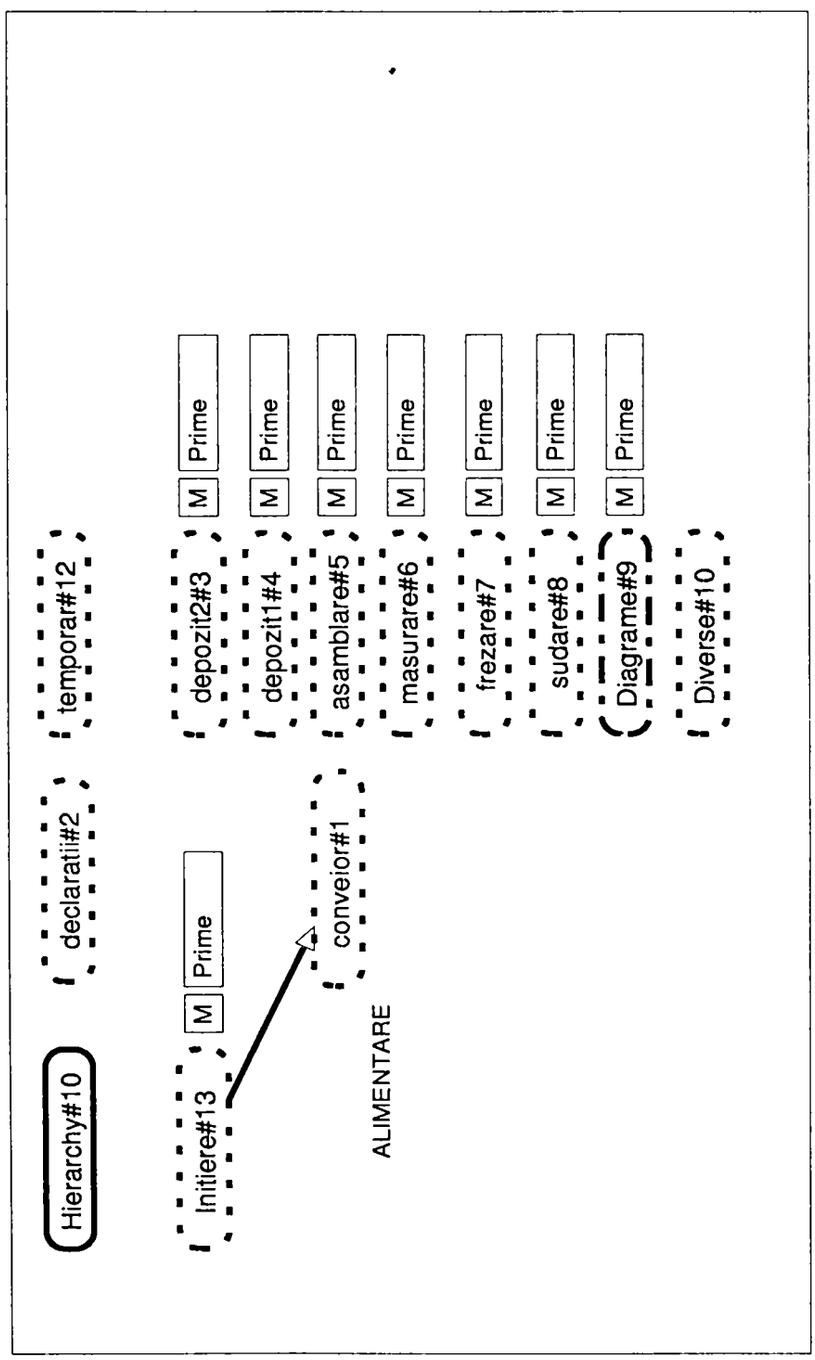
funcționarea celulei de fabricație în ciclu automat şi pot constitui obiectul unor studii ulterioare.

In concluzie, se poate afirma că optimizarea valorilor parametrilor cantitativi ai funcționării sistemului constituie un aspect important al optimizării fluxurilor de materiale.

3.6. Anexe

Anexa 3-1 Modelul DesignCPN pentru sistemul de fabricație în *varianta de transfer bazată pe partajarea resurselor*

Ierarhia sistemului



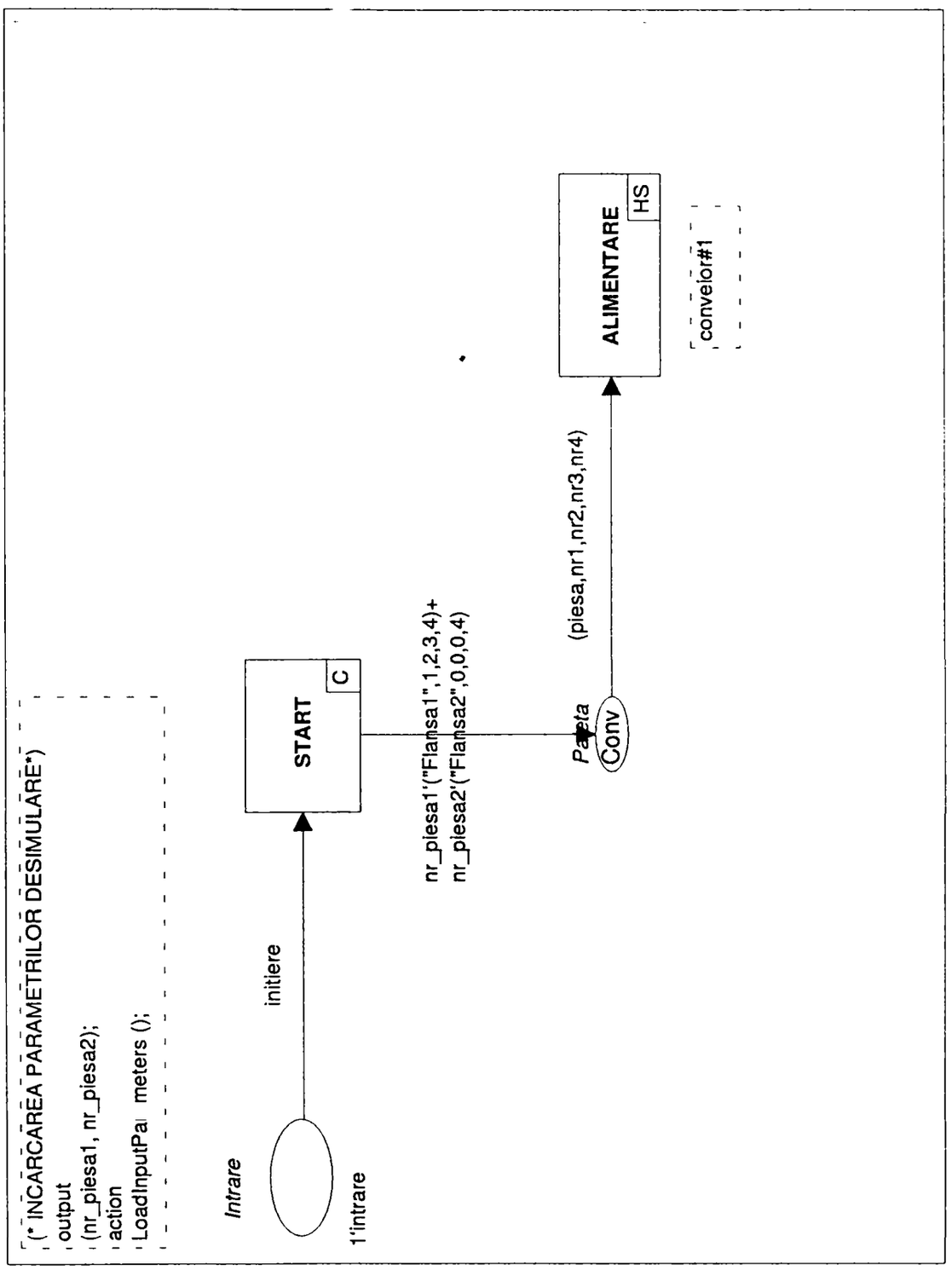
Declarații

```
color Piesa = string timed;
color Nr1 = int with 0..4 timed;
color Nr2 = int with 0..4 timed;
color Nr3 = int with 0..4 timed;
color Nr4 = int with 0..4 timed;
color Paleta = product Piesa * Nr1 * Nr2 * Nr3 * Nr4;
color Agv = Paleta;
color status = with disp timed;
color mf = Paleta;
color cloos = Paleta;
color mas = Paleta;
color depozit1 = Paleta;
color depozit2 = Paleta;
color Contr = with Flansa1 | Flansa2 | Ansamblu | Rebut | Alte | Frezare | Sudare |
Masurare ;
color INT = int;
color Intrare = with intrare;
color Validare= with frezare | sudare | dep1 | dep2 | asamblare | liber timed;

var valid: Validare;
var contr: Contr ms;
var piesa: Piesa;
var R1, R2, om: status;
var nr1: Nr1;
var nr2: Nr2;
var nr3: Nr3;
var nr4: Nr4;
var cnt, nr_piesa1, nr_piesa2, t_frezare, t_sudare, i: INT;
var initiere: Intrare;
var intr: Paleta ;
color rand0 = int with 1..100;
(*color rand1 = int with 1..100;*)
var s: rand0;
(*var r: rand1;*)

(*Functia care controleaza nivelul de rebuturi*)

use "/usr/local/DesignCPN/InputOutputLib.sml";

fun Ok(s:rand0) =

let

(* DESCHIDEREA FISIERULUI DE DATE *)
val infile = open_in "/home/mircea/Valori_initiale";

(* CITIREA VALORII PROCENTAJULUI DE PIESE BUNE *)
val skip_comments = (skiplab (infile); skip(infile); skiplab(infile); skiplab(infile)
);
val bun = (skip (infile); getint (infile));

(* INCHIDEREA FISIERULUI DE DATE *)
val close_file = (close_in infile);
in s<=bun
end;


(*Variabile statistice*)
val sv_AGV = SV'createint ();
val sv_Fre = SV'createint ();
val sv_Sud = SV'createint ();
val sv_Mas = SV'createint ();
val sv_Fin = SV'createint ();
val sv_Reb = SV'createint ();
val sv_Fl1 = SV'createint ();
val sv_Fl2 = SV'createint ();

(*Variabile de referinta*)
val old_AGV = ref 0;
val old_Fre = ref 0;
val old_Sud = ref 0;
val old_Mas = ref 0;
val old_Fin = ref 0;
val old_Reb = ref 0;
val old_Fl1 = ref 0;
val old_Fl2 = ref 0;

(*Variabila de stocare a timpului de sistem si a numarului de pasi*)

val Pas=step;
val Time=time;
```

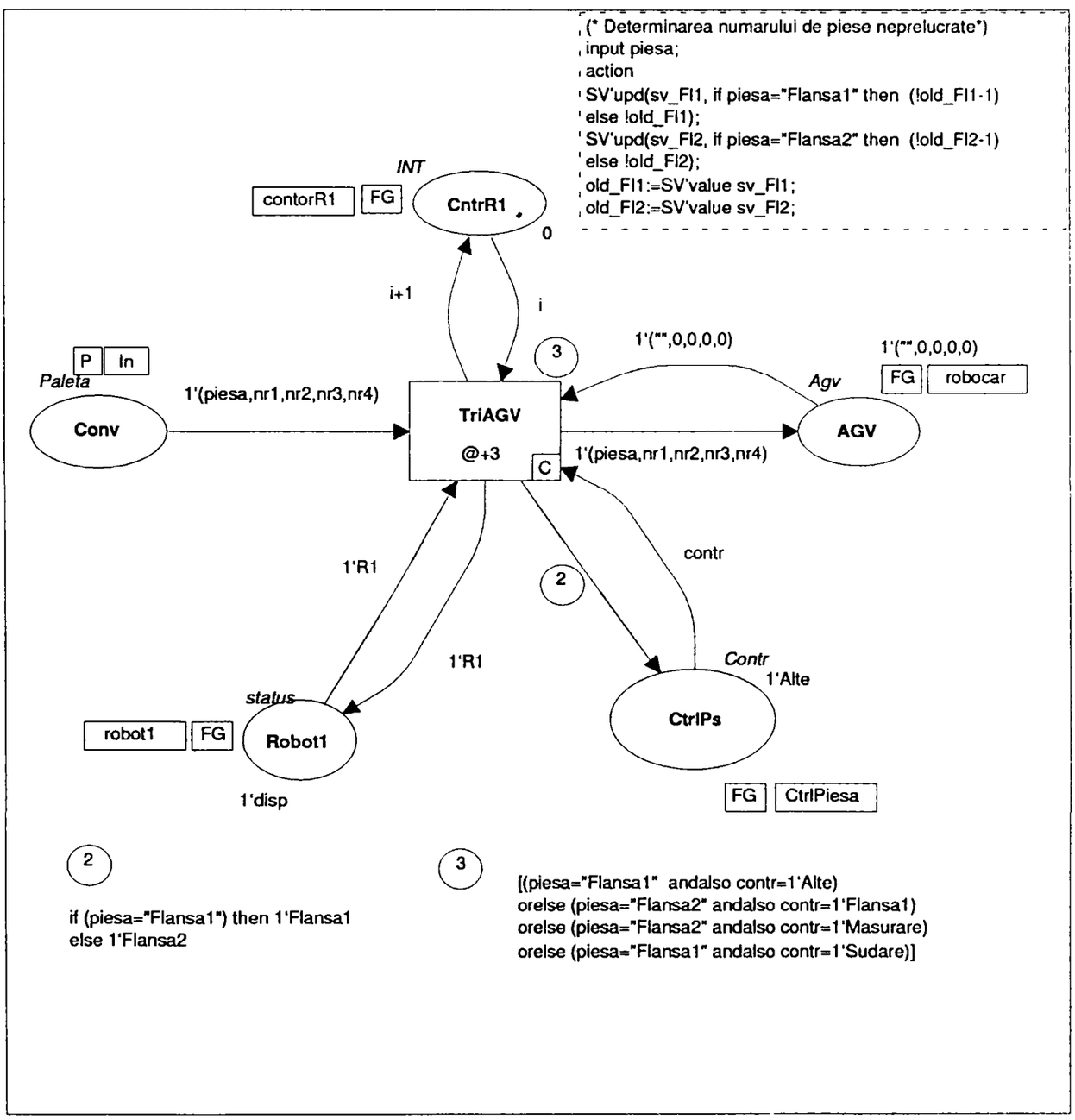
Temporar

```
val par = !;
val OutText=ref "";
(* CITIREA VALORILOR PARAMETRILOR DE INCARCARE AI CONVEIORULUI *)

use "/home/DesignCPN/TutorialDiagrams/InputOutputL ).sml";
fun LoadInputParameters () =
let

(* DESCHIDEREA FISIERULUI DE DATE *)
val infile = open_in "/home/mircea/Valori_initiale";
(*(DSFile_NameDialog
                {prompt="Selecteaza fisierul de intrare: Valori_initiale",
okButtonLabel="Da-i!", path="/home/mircea/", writeBox=false});*)

(* CITIREA DATELOR REFERITOARE LA NUMARUL INITIAL DE PIESE *)
val skip_comments = (skiplab (infile); skip (infile); skiplab (infile));
val nr_piesa1 = (skip (infile); getint (infile));
val nr_piesa2 = (skip (infile); getint (infile));

(*INITIALIZAREA VARIABILELOR STATISTICE REFERITOARE LA NUMARUL DE PIESE*)
val nr_de_piese = (old_F11:=nr_piesa1;
     old_F12:=nr_piesa2);

val initial_sv = (SV'upd(sv_F11, !old_F11);
     SV'upd(sv_F12, !old_F12));


(* INCHIDEREA FISIERULUI DE DATE *)
val close_file = (close_in infile);

(*PARAMETRII CARE VOR FI ASOCIATI CU VARIABILELE DE PE ARCELE DE IESIRE*)
in
(nr_piesa1, nr_piesa2)
end;
(*
(* OPERATIA DE VALIDARE A UNUI TRASEU DE TRANSPORT*)

fun Selectie (piesa:string, nr4:int, contr:Contr ms) =

let
val valid= (if (piesa="Flansa1") andalso (contr=1`Flansa1)
   then frezare
else if (piesa="Flansa2") andalso (contr=1`Masurare)
   then asamblare
else if (piesa="Ansamblu") andalso (contr=1`Ansamblu)
   then sudare
else if (piesa="Ansamblu") andalso (nr4=0) andalso (contr=1`Sudare)
   then dep1
else if (piesa="Rebut") andalso (contr=1`Rebut)
   then dep2
else liber)
in valid
end;
*)
(* FUNCTIA PENTRU SCRIEREA PARAMETRILOR DE IESIRE AI SISTEMULUI*)
fun Scrievalstat () =
let
val outfile = open_append "/home/mircea/date_ies"
in
(output (outfile, "************************************************** \n");
(*output (outfile, (!OutputText));*)
output (outfile, " \n" );
output (outfile,"Miscari AGV= "^makestring(par old_AGV+1)^" La momentul T=
"^makestring(Time ()));
output (outfile, "\n \n");
output (outfile, "Piese frezate= "^makestring(par old_Fre+1));
output (outfile, "\n \n");
output (outfile, "Piese sudate= "^makestring(par old_Sud+1));
output (outfile, "\n \n");
output (outfile, "Piese masurate= "^makestring(par old_Mas+1));
output (outfile, "\n \n");
output (outfile, "Piese finite= "^makestring(par old_Fin+1));
output (outfile, "\n \n");
output (outfile, "Piese rebut= "^makestring(par old_Reb + SV'value sv_Reb));
output (outfile, "\n \n");
output (outfile, "Productivitatea celulei= "^makestring(real(par old_Fin+1)/real(
Time()))^" Piese finite/minut");
output (outfile, "\n \n");
output (outfile, "Timpul mediu de prelucrare= "^makestring(real(Time())/real(!
old_Fin+1))^" minute/piesa");
output (outfile, "\n \n");
output (outfile, "Numarul de piese tip 1 in ciclu= "^makestring(SV'max sv_F11 -
SV'value sv_F11 - SV'sum sv_Fin - SV'sum sv_Reb));
output (outfile, "\n \n");
output (outfile, "Numarul de piese tip 2 in ciclu= "^makestring(SV'max sv_F12 -
SV'value sv_F12 - SV'sum sv_Fin));
output (outfile, "\n \n");
output (outfile, "Numarul de piese tip 1 neprelucrate= "^makestring(SV'value sv_F11))
;
output (outfile, "\n \n");
output (outfile, "Numarul de piese tip 2 neprelucrate= "^makestring(SV'value sv_F12))
;
output (outfile, "\n \n");
output (outfile, "\n");

close_out (outfile))
end;

(*****************************************************)
```

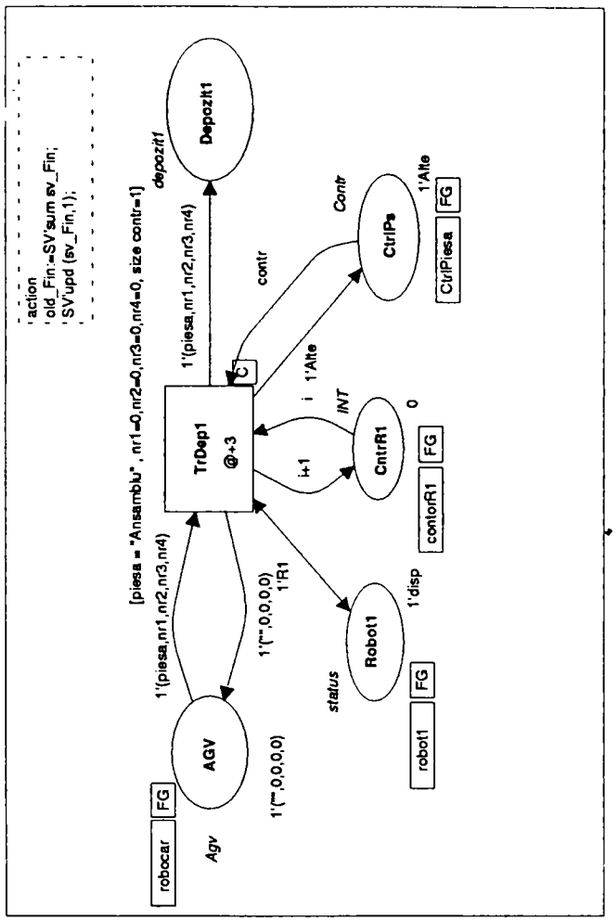
Initiere



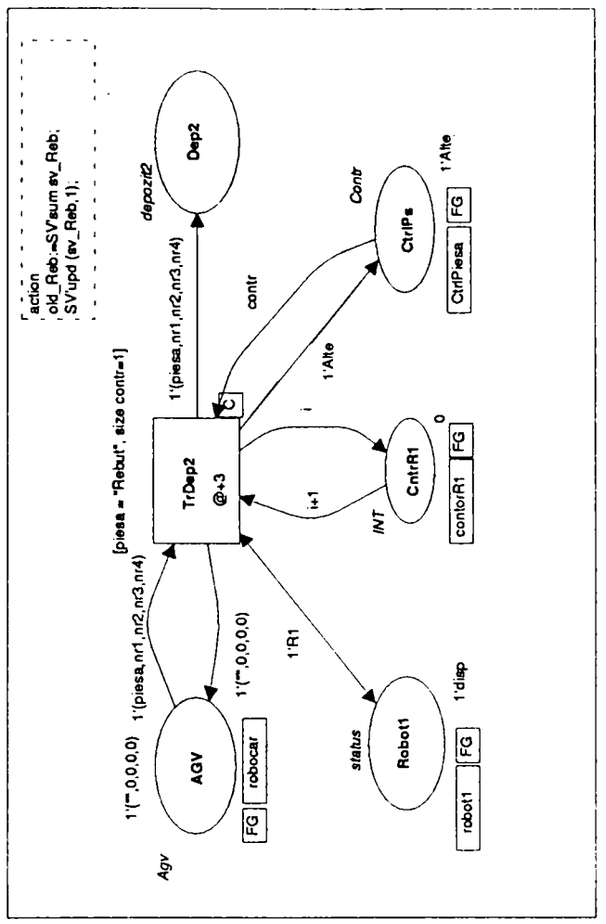
Conveior



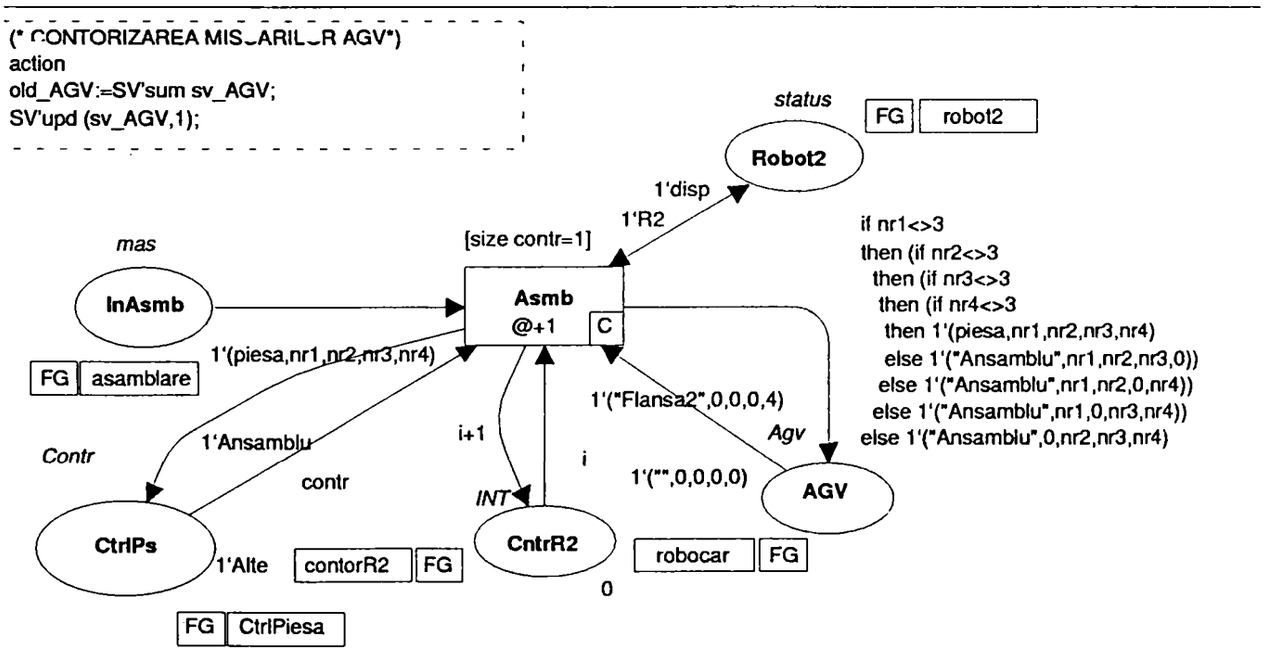
Depozit1



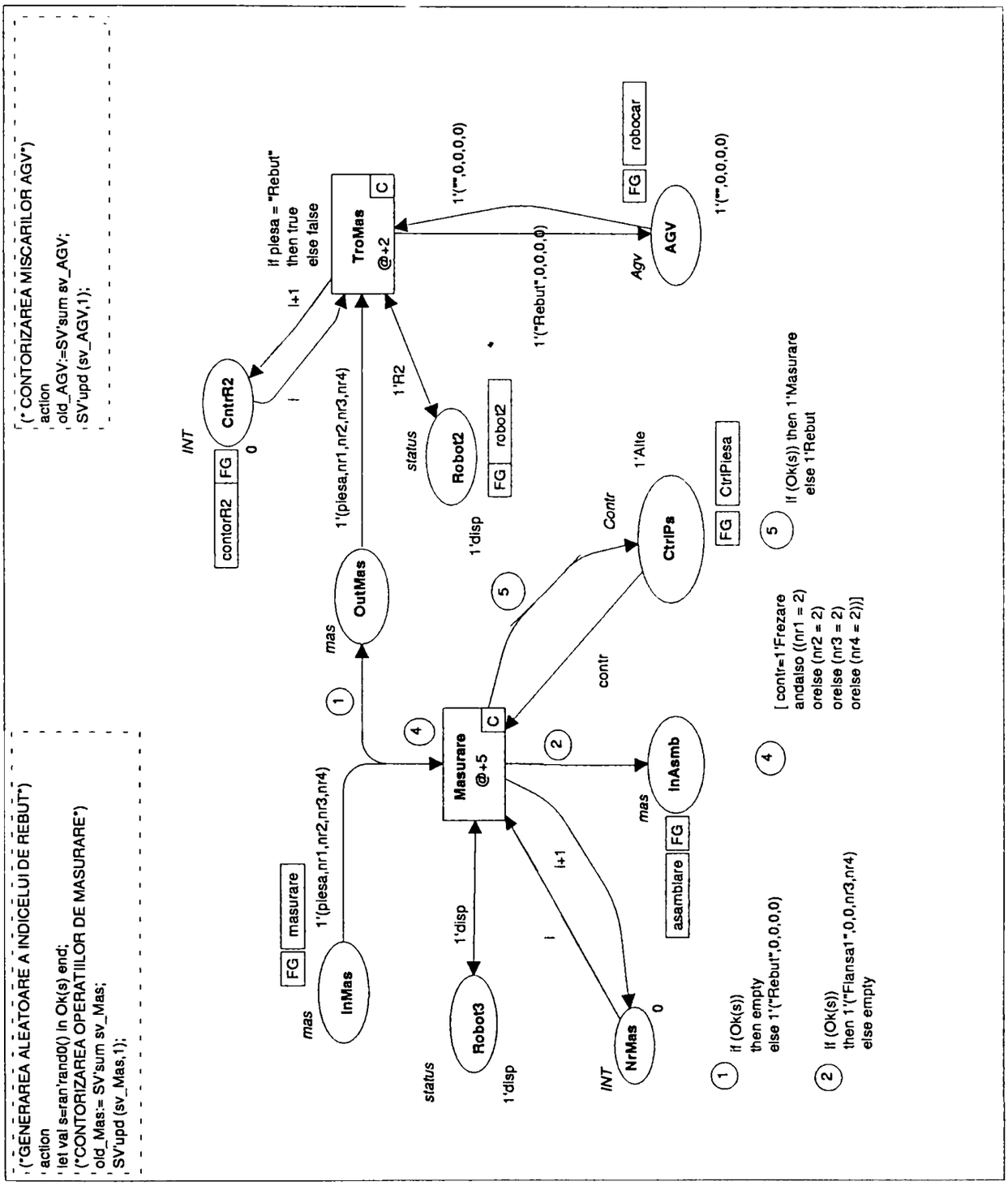
Depozit2



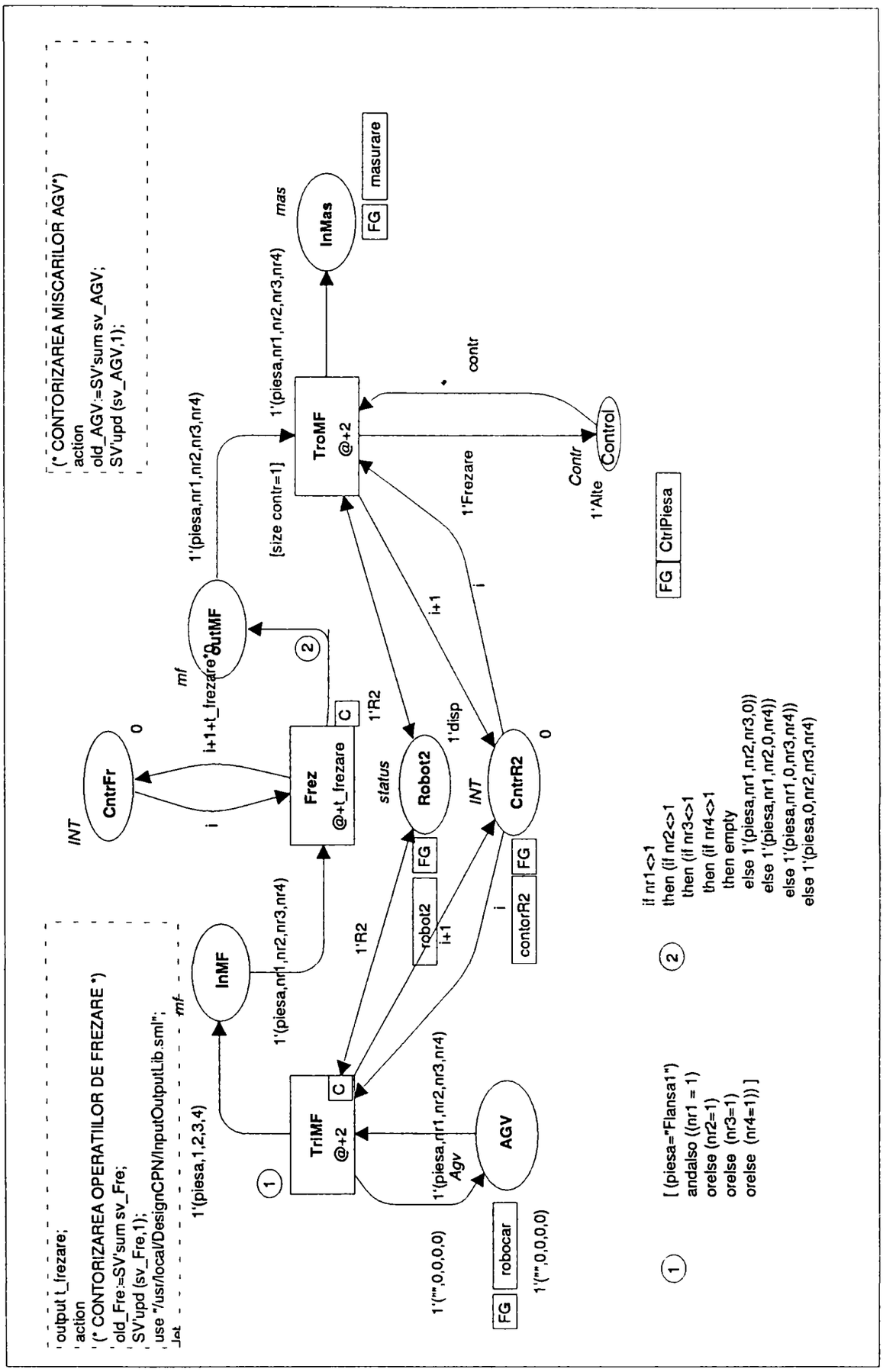
Asamblare



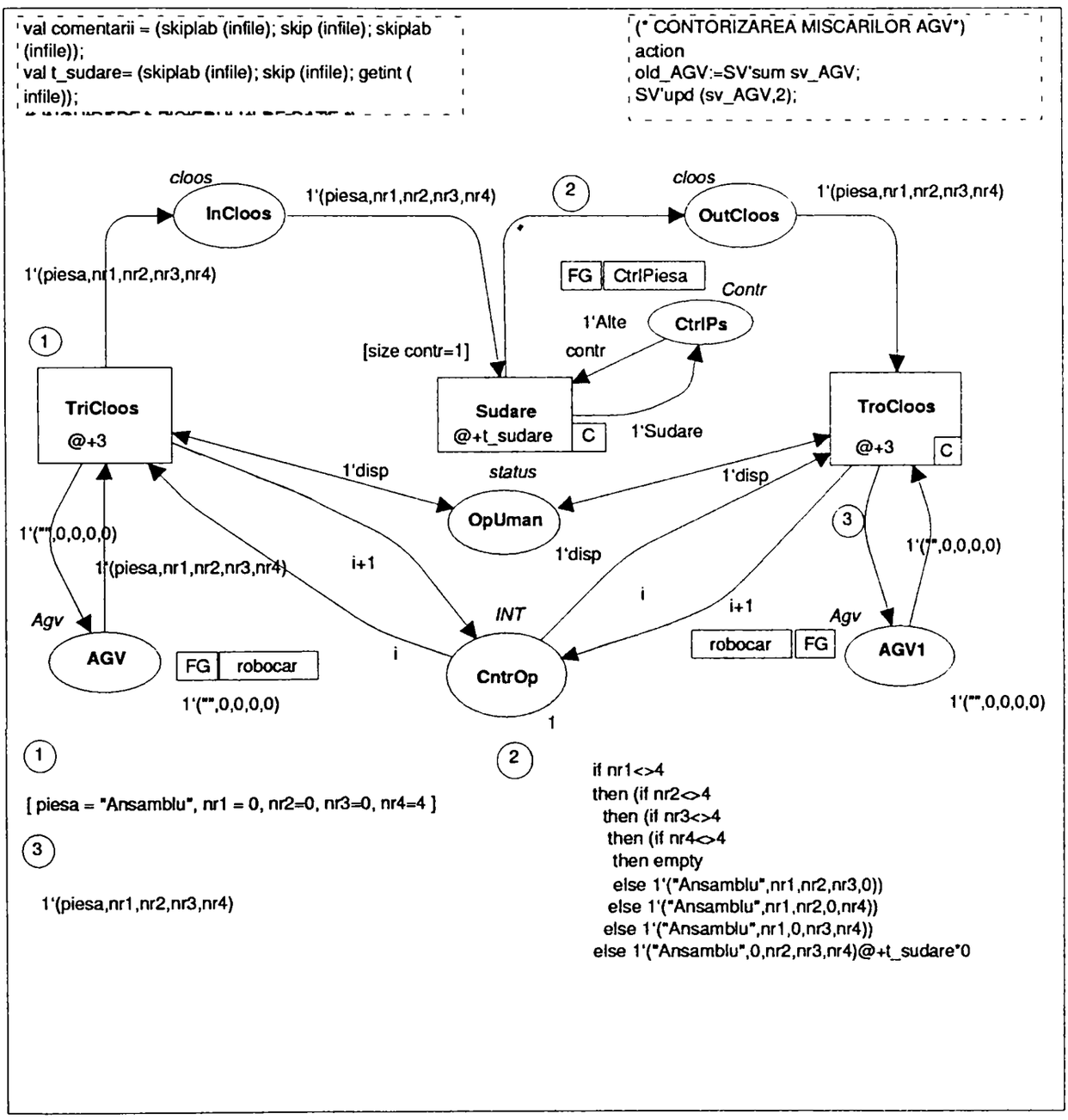
Măsurare



Frezare

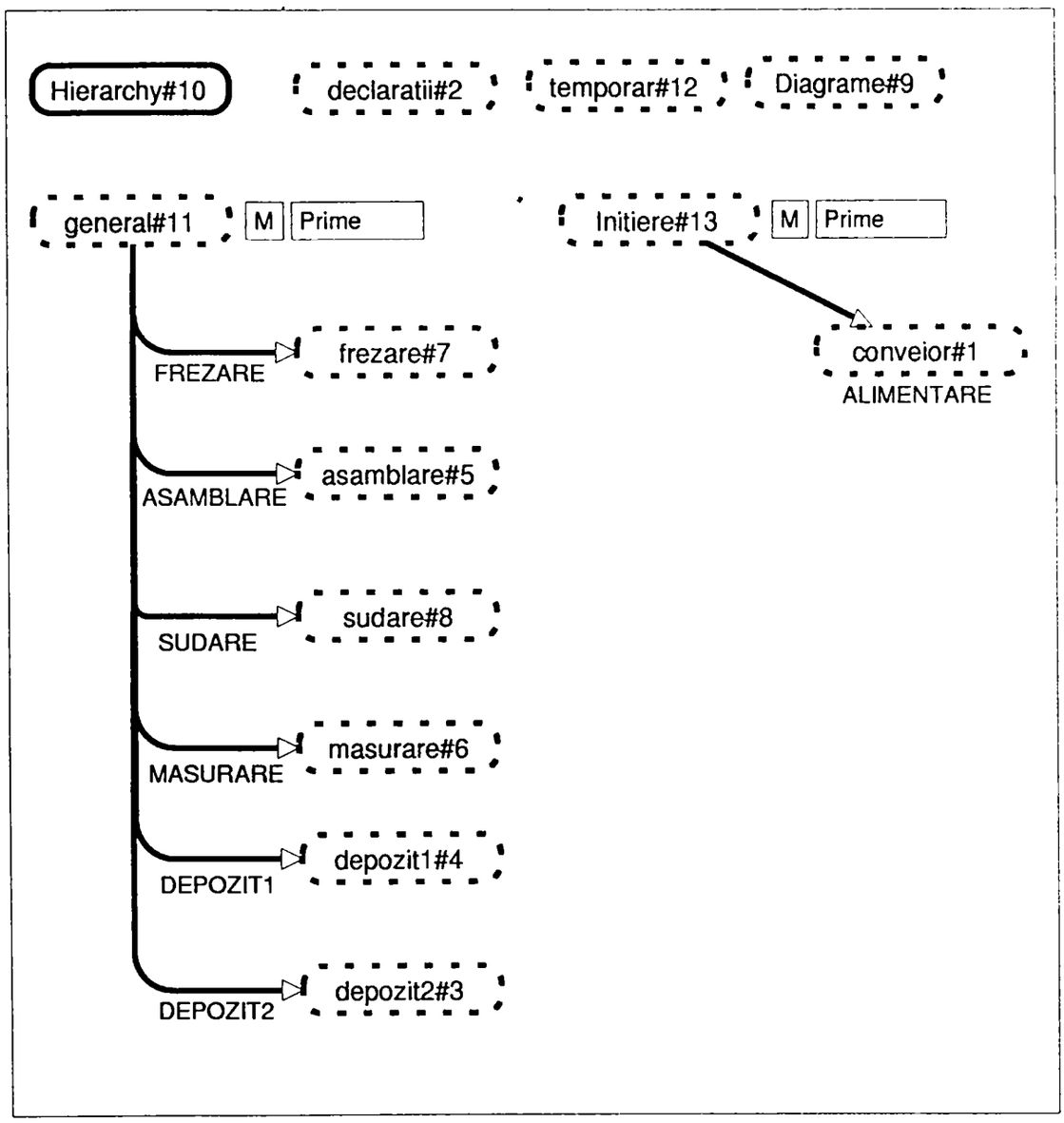


Sudare


val comentarii = (skiplab (infile); skip (infile); skiplab (infile));
val t_sudare= (skiplab (infile); skip (infile); getint (infile));
(* CONTORIZAREA MISCARILOR AGV*)
action
old_AGV:=SV'sum sv_AGV;
SV'upd (sv_AGV,2);
cloos
InCloos
1'(piesa,nr1,nr2,nr3,nr4)
2
cloos
OutCloos
1'(piesa,nr1,nr2,nr3,nr4)
1'(piesa,nr1,nr2,nr3,nr4)
FG
CtrlPiesa
Contr
1'Alte
CtrlPs
1
[size contr=1]
contr
TriCloos
@+3
Sudare
@+t_sudare
C
1'Sudare
TroCloos
@+3
C
1'disp
status
1'disp
OpUman
1'("",0,0,0,0)
1'disp
3
1'("",0,0,0,0)
1'(piesa,nr1,nr2,nr3,nr4)
i+1
i
INT
i+1
Agv
Agv
AGV
FG
robocar
i
CntrOp
robocar
FG
AGV1
1'("",0,0,0,0)
1'("",0,0,0,0)
1
1
2
[piesa = "Ansamblu", nr1 = 0, nr2=0, nr3=0, nr4=4]
3
1'(piesa,nr1,nr2,nr3,nr4)
if nr1<>4
then (if nr2<>4
then (if nr3<>4
then (if nr4<>4
then empty
else 1'("Ansamblu",nr1,nr2,nr3,0))
else 1'("Ansamblu",nr1,nr2,0,nr4))
else 1'("Ansamblu",nr1,0,nr3,nr4))
else 1'("Ansamblu",0,nr2,nr3,nr4)@+t_sudare*0

Anexa 3-2 Modelul DesignCPN pentru sistemul de fabricație în varianta de *transfer bazat pe tehnologia de fabricatie*

Ierarhia sistemului



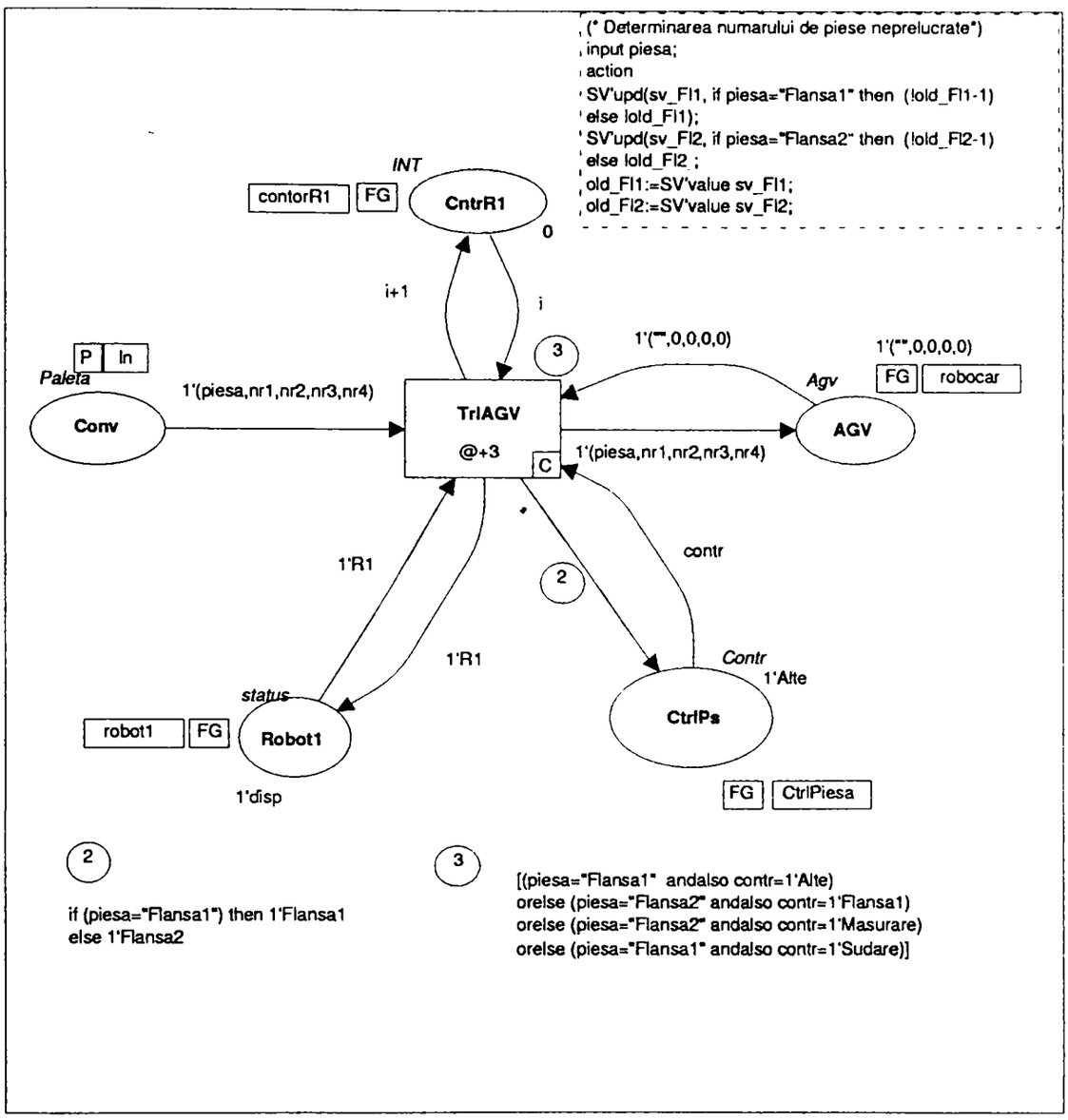
Declaratii

```
color Piesa = string timed;
color Nr1 = int with 0..4 timed;
color Nr2 = int with 0..4 timed;
color Nr3 = int with 0..4 timed;
color Nr4 = int with 0..4 timed;
color Paleta = product Piesa * Nr1 * Nr2 * Nr3 * Nr4;
color Agv = Paleta;
color status = with disp timed;
color mf = Paleta;
color cloos = Paleta;
color mas = Paleta;
color depozit1 = Paleta;
color depozit2 = Paleta;
color Contr = with Flansa1 | Flansa2 | Ansamblu | Rebut | Alte | Frezare |
Sudare | Masurare ;
color INT = int;
color Intrare = with intrare;
color Validare= with frezare | sudare | dep1 | dep2 | asamblare | liber timed;

var valid: Validare;
var contr: Contr ms;
var piesa: Piesa;
var R1, R2, om: status;
var nr1: Nr1;
var nr2: Nr2;
var nr3: Nr3;
var nr4: Nr4;
var Pas, Time, cnt, bun, nr_piesa1, nr_piesa2, t_frezare, t_sudare, i: INT;
var initiere: Intrare;
var intr: Paleta ;
color rand0 = int with 1..100;
(*color rand1 = int with 1..100;*)
var s: rand0;
(*var r: rand1;*)

(*Functia care controleaza nivelul de rebuturi*)

use "/usr/local/DesignCPN/InputOutputLib.sml";

fun Ok(s:rand0) =

let

(* DESCHIDEREA FISIERULUI DE DATE *)
val infile = open_in "/home/mircea/Valori_initiale";

(* CITIREA VALORII PROCENTAJULUI DE PIESE BUNE *)
val skip_comments = (skiplab (infile); skip(infile); skiplab(infile); skiplab(
infile));
val bun = (skip (infile); getint (infile));

(* INCHIDEREA FISIERULUI DE DATE *)
val close_file = (close_in infile);

(*PARAMETRII CARE VOR FI ASOCIATI CU VARIABILELE DE PE ARCELE DE IESIRE*)

in s<=bun
end;


(*Variabile statistice*)
val sv_AGV = SV'createint ();
val sv_Fre = SV'createint ();
val sv_Sud = SV'createint ();
val sv_Mas = SV'createint ();
val sv_Fin = SV'createint ();
val sv_Reb = SV'createint ();
val sv_F11 = SV'createint ();
val sv_F12 = SV'createint ();

(*Variabile de referinta*)
val old_AGV = ref 0;
val old_Fre = ref 0;
val old_Sud = ref 0;
val old_Mas = ref 0;
val old_Fin = ref 0;
val old_Reb = ref 0;
val old_F11 = ref 0;
val old_F12 = ref 0;

(*Variabila de stocare a timpului de sistem*)

val Pas=step;
val Time=time;
```

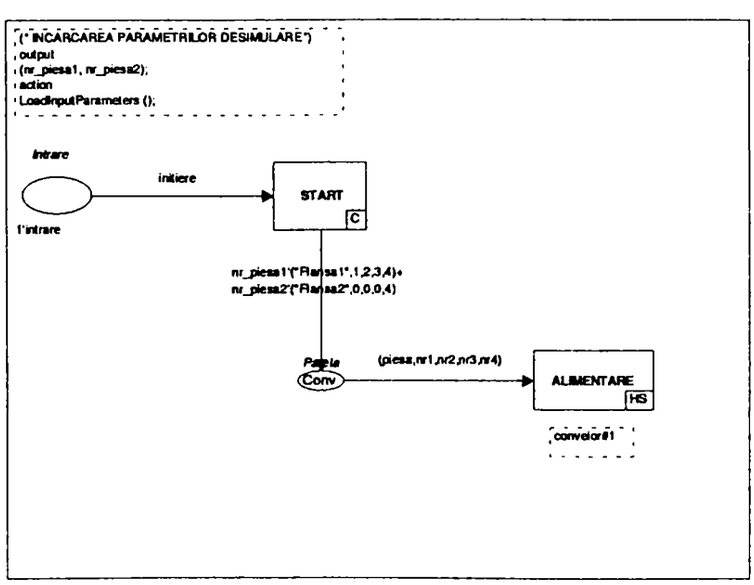
Temporar

```
val par = 1;

(* CITIREA VALORILOR PARAMETRILOR DE INCARCARE AI CONVEIORULUI *)

use "/usr/local/DesignCPN/InputOutputLib.sml";
fun LoadInputParameters () =
let

(* DESCHIDEREA FISIERULUI DE DATE *)
val infile = open_in "/home/mircea/Valori_initiale";
    (*open_in (DSFile.NameDialog
                 {prompt="Select Input File, Valori_initiale", okButtonLabel="De-a-l",
                  path="/home/mircea/", writeBox=false});*)

(* READ JOB STREAM PARAMETERS FROM THE PARAMETERS FILE *)
val skip_comments = (skiplab (infile); skip(infile); skiplab(infile));
val nr_piesa1 = (skip (infile); getint (infile));
val nr_piesa2 = (skip (infile); getint (infile));
(*INITIALIZAREA VARIABILELOR STATISTICE REFERITOARE LA NUMARUL DE PIESE*)
val nr_de_piese = (old_F11:=nr_piesa1;
     old_F12:=nr_piesa2);

val initial_sv = (SV'upd(sv_F11, !old_F11);
     SV'upd(sv_F12, !old_F12));

(* INCHIDEREA FISIERULUI DE DATE *)
val close_file = (close_in infile);

(*PARAMETRII CARE VOR FI ASOCIATI CU VARIABILELE DE PE ARCELE DE IESIRE*)
in
(nr_piesa1, nr_piesa2)
end;


(* OPERATIA DE VALIDARE A UNUI TRASEU DE TRANSPORT*)

fun Selectie (piesa:string, nr4:int, contr:Contr ms) =

let
val valid= (if (piesa="Flansa1") andalso (contr=1`Flansa1)
   then frezare
else if (piesa="Flansa2") andalso (contr=1`Masurare)
   then asamblare
else if (piesa="Ansamblu") andalso (contr=1`Ansamblu)
   then sudare
else if (piesa="Ansamblu") andalso (nr4=0) andalso (contr=1`Sudare)
   then dep1
else if (piesa="Rebut") andalso (contr=1`Rebut)
   then dep2
else liber)
in valid
end;

(* FUNCTIA PENTRU SCRIEREA PARAMETRILOR DE IESIRE AI SISTEMULUI*)
fun Scrievsistat () =
let
val outfile = open_append "/home/mircea/date_ies"
in
(output (outfile, "************************************************** \n");
(*output (outfile, ((OutputText)));*)
output (outfile, " \n" );
output (outfile,"Miscari AGV= "^makestring(par old_AGV+1)^" La momentul T=
"^makestring(Time ())^
output (outfile, "\n \n");
output (outfile, "Piese frezate= "^makestring(par old_Fre+1));
output (outfile, "\n \n");
output (outfile, "Piese sudate= "^makestring(par old_Sud+1));
output (outfile, "\n \n");
output (outfile, "Piese masurate= "^makestring(par old_Mas+1));
output (outfile, "\n \n");
output (outfile, "Piese finite= "^makestring(par old_Fin+1));
output (outfile, "\n \n");
output (outfile, "Piese rebut= "^makestring(par old_Reb + SV'value sv_Reb));
output (outfile, "\n \n");
output (outfile, "Productivitatea celulei= "^makestring(real(par old_Fin+1)/real(Time(
))))^" Piese finite/minut");
output (outfile, "\n \n");
output (outfile, "Timpul mediu de prelucrare= "^makestring(real(Time())/real(!
old_Fin+1))^" minute/piese");
output (outfile, "\n \n");
output (outfile, "Numarul de piese tip 1 in ciclu= "^makestring(SV'max sv_F11 -
SV'value sv_F11 - SV'sum sv_Fin - SV'sum sv_Reb));
output (outfile, "\n \n");
output (outfile, "Numarul de piese tip 2 in ciclu= "^makestring(SV'max sv_F12 -
SV'value sv_F12 - SV'sum sv_Fin));
output (outfile, "\n \n");
output (outfile, "Numarul de piese tip 1 neprelucrate= "^makestring(SV'value sv_F11));
output (outfile, "\n \n");
output (outfile, "Numarul de piese tip 2 neprelucrate= "^makestring(SV'value sv_F12));
output (outfile, "\n \n");
output (outfile, "\n");

close_out (outfile)
end;
```

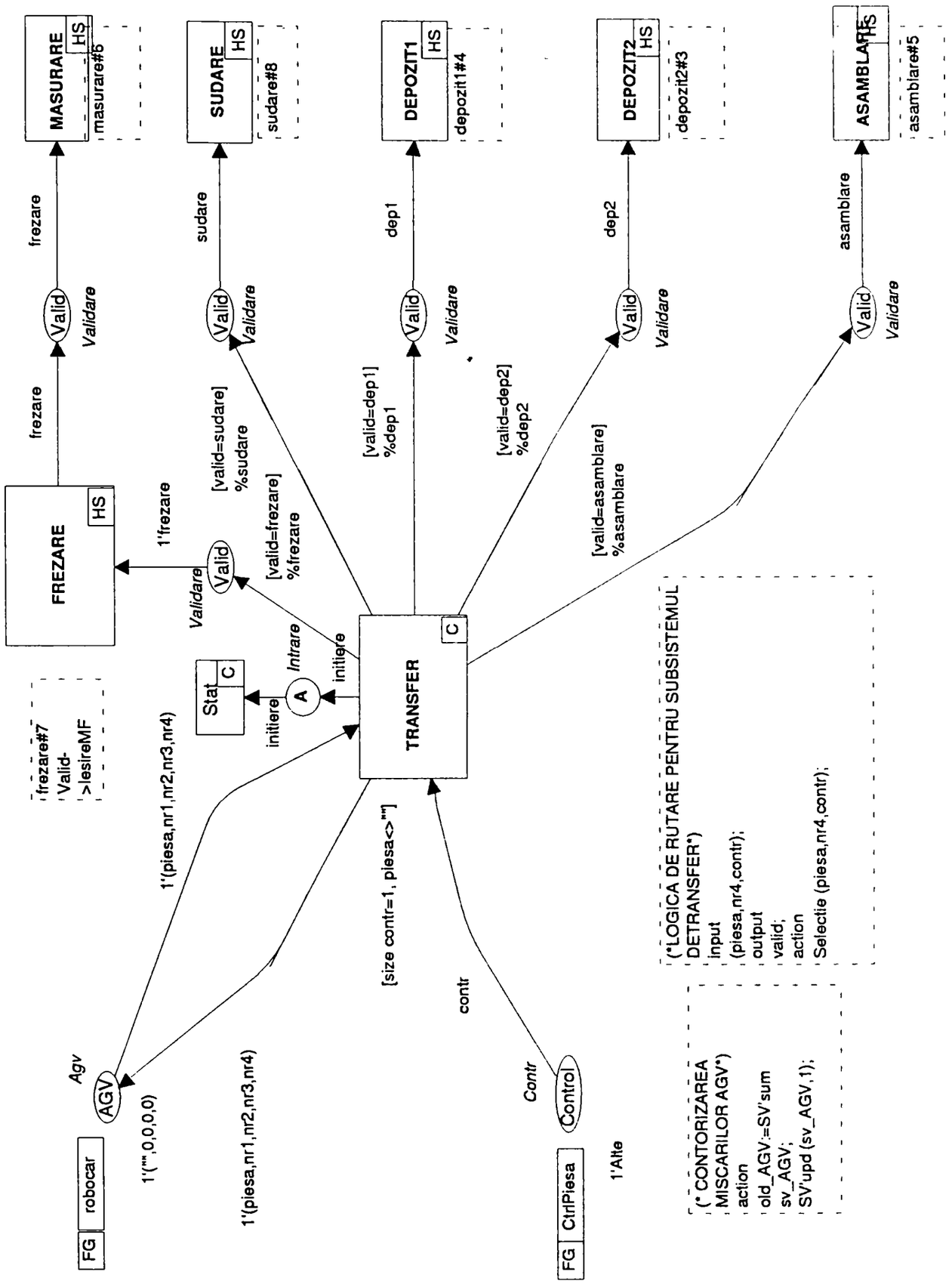
Conveior



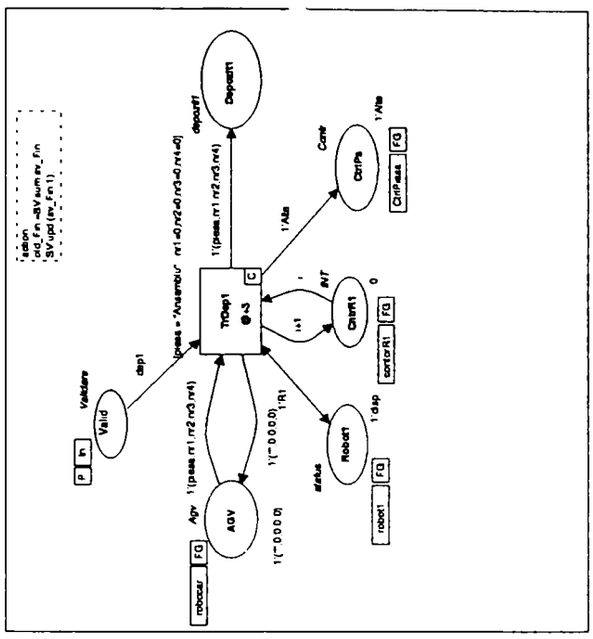
Initiere



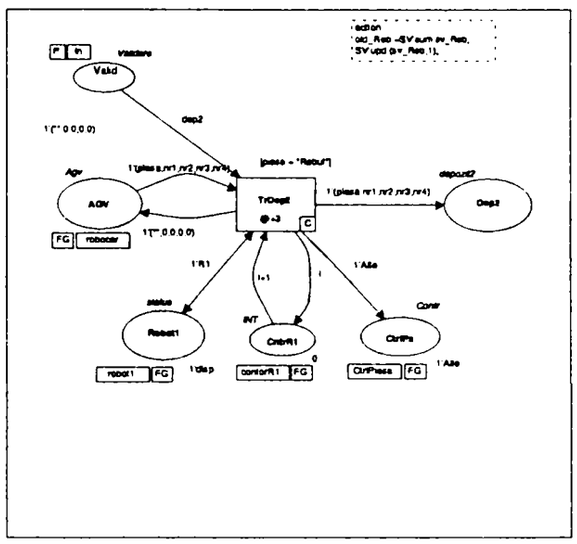
General



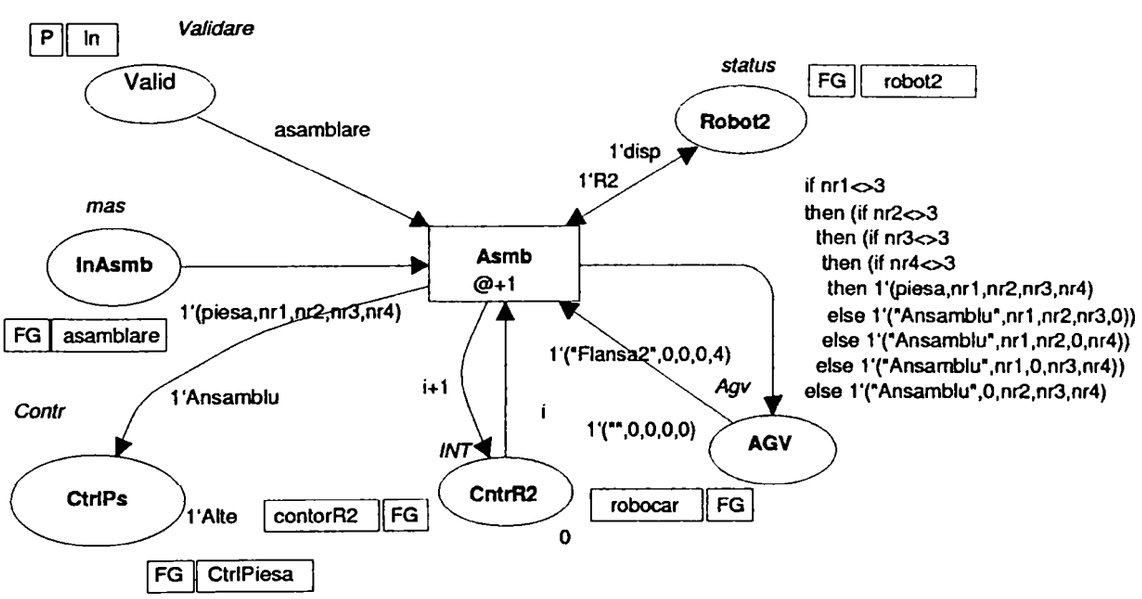
Depozit1



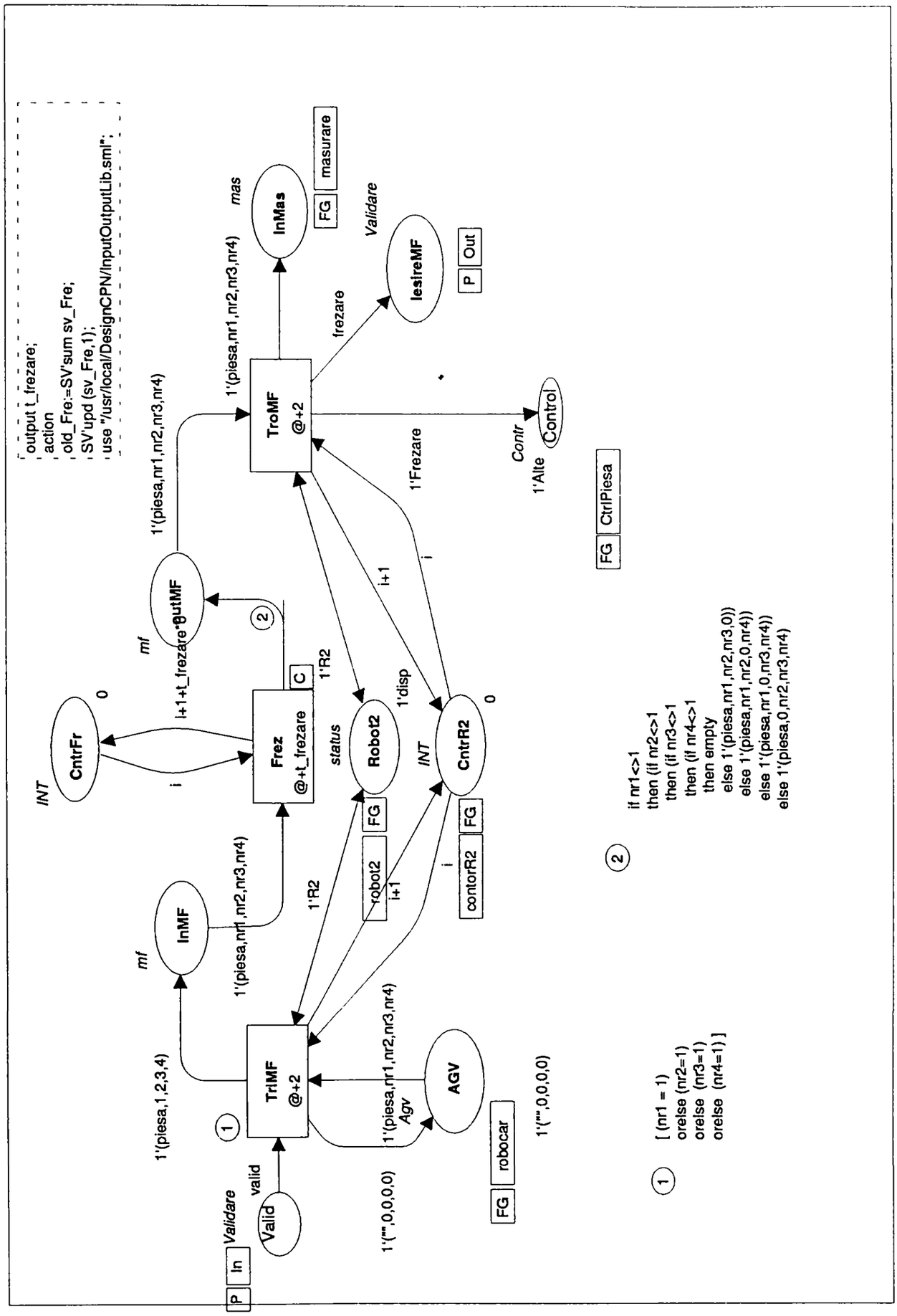
Depozit2



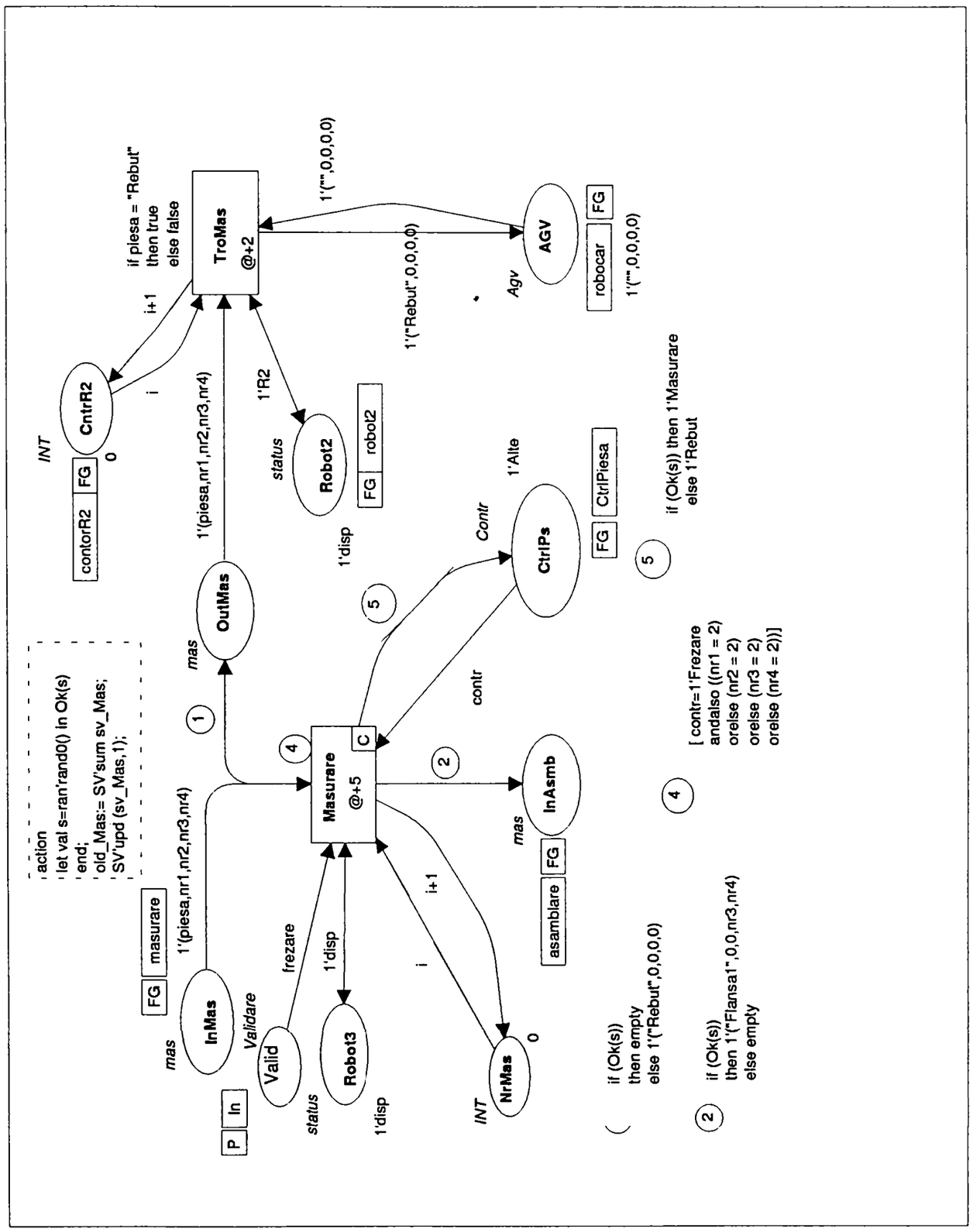
Asamblare



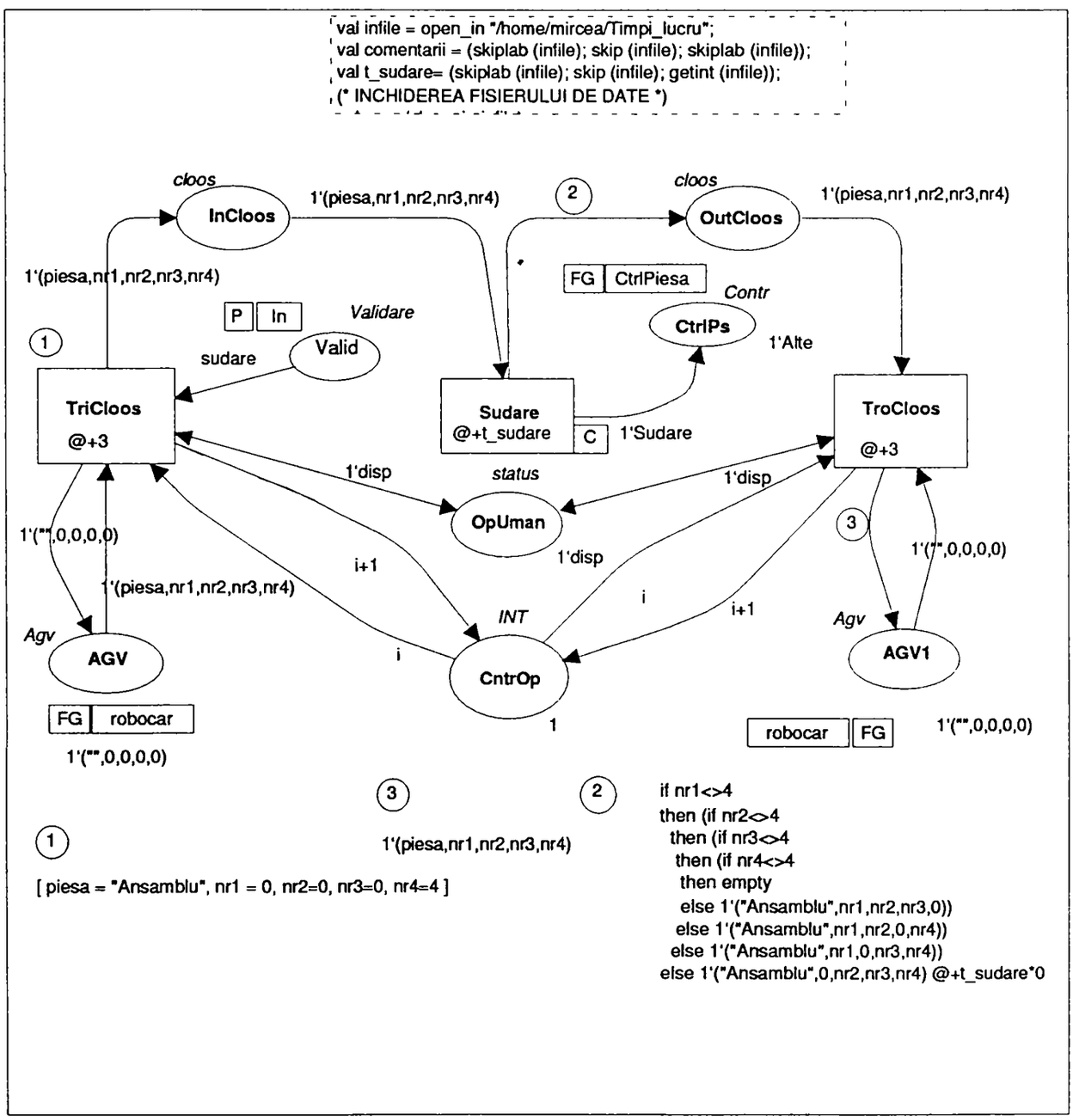
Frezare



Măsurare



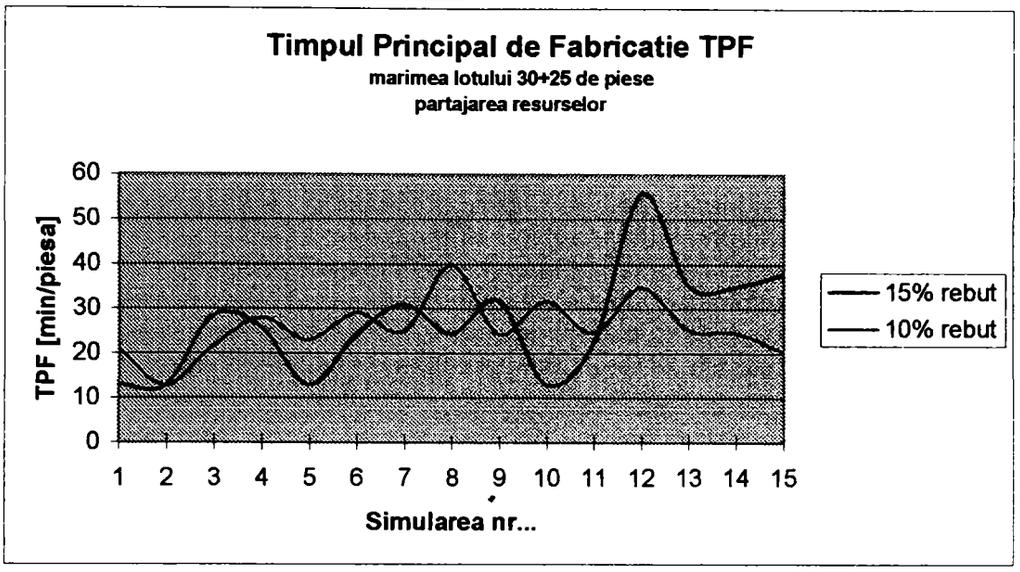
Sudare

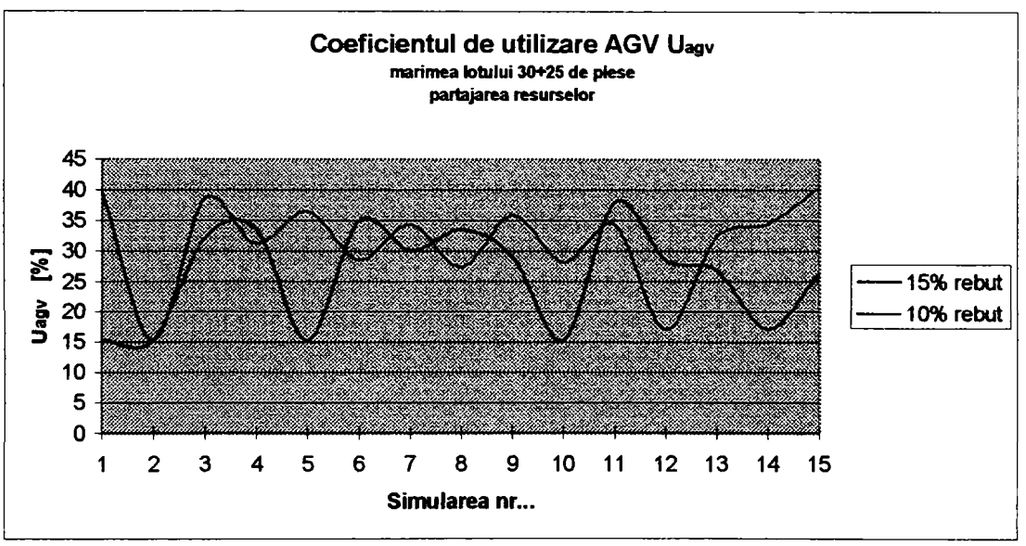

val infile = open_in "/home/mircea/Timpi_lucru";
val comentarii = (skiplab (infile); skip (infile); skiplab (infile));
val t_sudare= (skiplab (infile); skip (infile); getint (infile));
(* INCHIDEREA FISIERULUI DE DATE *)
cloos
InCloos
1'(piesa,nr1,nr2,nr3,nr4)
2
cloos
OutCloos
1'(piesa,nr1,nr2,nr3,nr4)
1'(piesa,nr1,nr2,nr3,nr4)
FG
CtrlPiesa
Contr
CtrlPs
1'Alte
P
In
Validare
1
sudare
Valid
TriCloos
@+3
Sudare
@+t_sudare
C
1'Sudare
TroCloos
@+3
1'disp
status
1'disp
OpUman
1'("",0,0,0,0)
1'disp
3
1'("",0,0,0,0)
i+1
1'(piesa,nr1,nr2,nr3,nr4)
i
INT
i+1
Agv
AGV
i
CntrOp
Agv
AGV1
FG
robocar
1'("",0,0,0,0)
1
robocar
FG
1'("",0,0,0,0)
3
1'(piesa,nr1,nr2,nr3,nr4)
2
1
[piesa = "Ansamblu", nr1 = 0, nr2=0, nr3=0, nr4=4]
if nr1<>4
then (if nr2<>4
then (if nr3<>4
then (if nr4<>4
then empty
else 1'("Ansamblu",nr1,nr2,nr3,0))
else 1'("Ansamblu",nr1,nr2,0,nr4))
else 1'("Ansamblu",nr1,0,nr3,nr4))
else 1'("Ansamblu",0,nr2,nr3,nr4) @+t_sudare*0

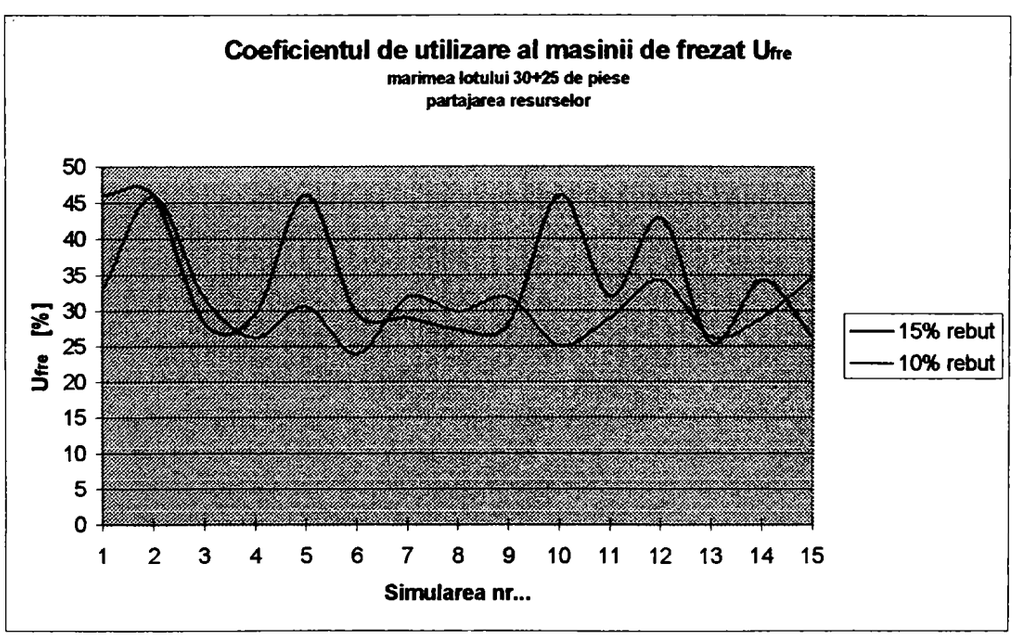
Fişier cu date de ieşire (fragment)

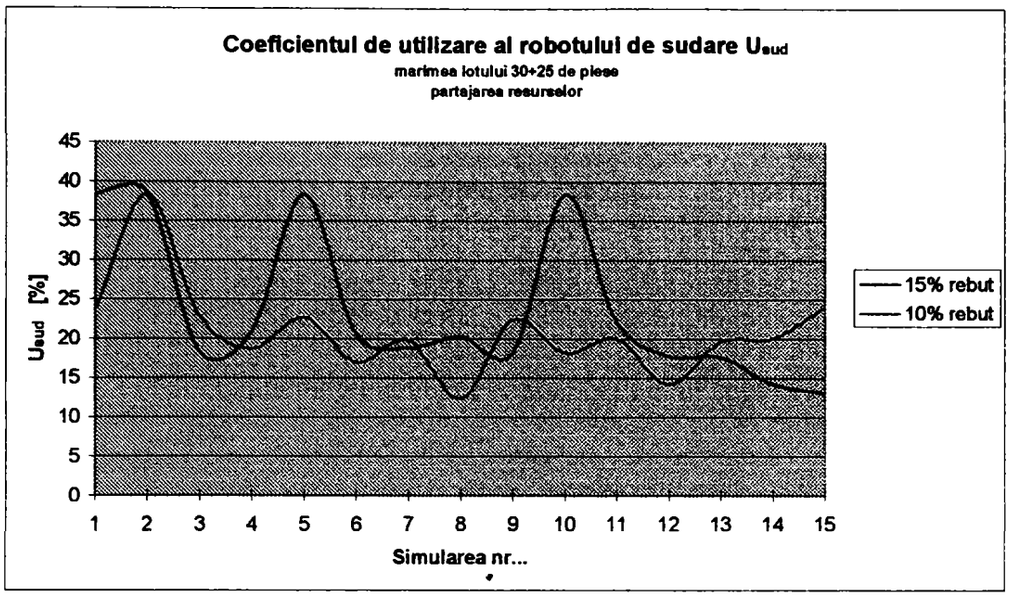
```
**************************************************
Miscari AGV= 215 La momentul T= 1708
Piese frezate= 58
Piese sudate= 50
Piese masurate= 58

Piese finite= 50

Piese rebut= 7

Productivitatea celulei= 0.0292740046838408 Piese finite/minut

Timpul mediu de prelucrare= 34.16 minute/piesa

Numarul de piese tip 1 in ciclu= 1

Numarul de piese tip 2 in ciclu= 0
 **************************************************

Miscari AGV= 213 La momentul T= 1687

Piese frezate= 57

Piese sudate= 50

Piese masurate= 57

Piese finite= 50

Piese rebut= 6

Productivitatea celulei= 0.02963841138115 Piese finite/minut

Timpul mediu de prelucrare= 33.74 minute/piesa

Numarul de piese tip 1 in ciclu= 1

Numarul de piese tip 2 in ciclu= 0
 **************************************************

Miscari AGV= 215 La momentul T= 1708

Piese frezate= 58

Piese sudate= 50

Piese masurate= 58

Piese finite= 50

Piese rebut= 7

Productivitatea celulei= 0.0292740046838408 Piese finite/minut

Timpul mediu de prelucrare= 34.16 minute/piesa

Numarul de piese tip 1 in ciclu= 1
```

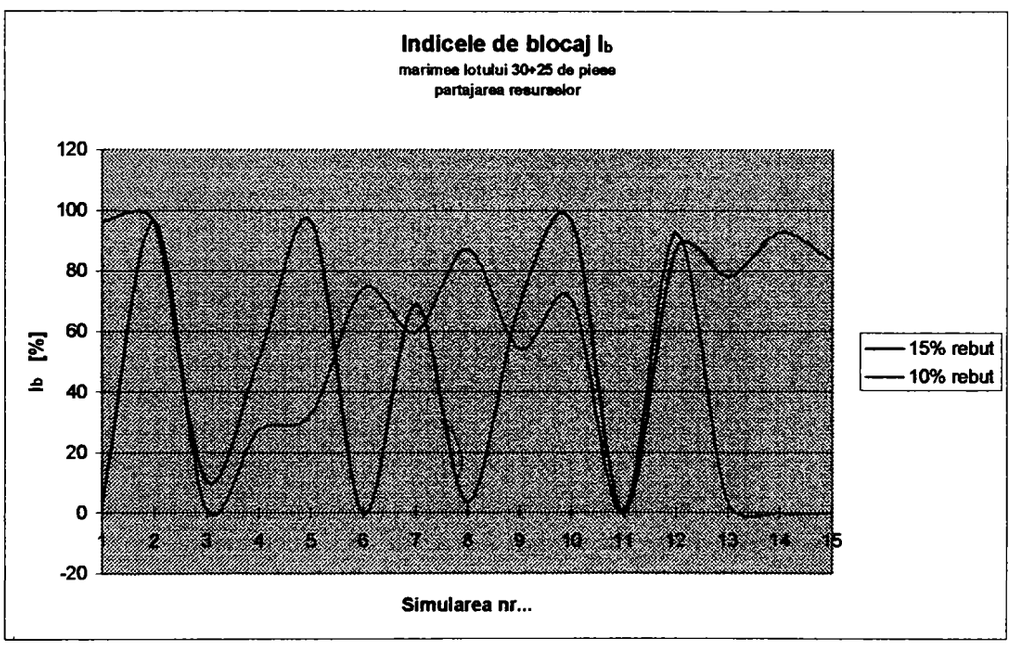
Anexa 3-3 Diagramele de variaţie a indicilor de performanţă pentru *lotul 30+25* şi *logica de transfer bazată pe partajarea resurselor*

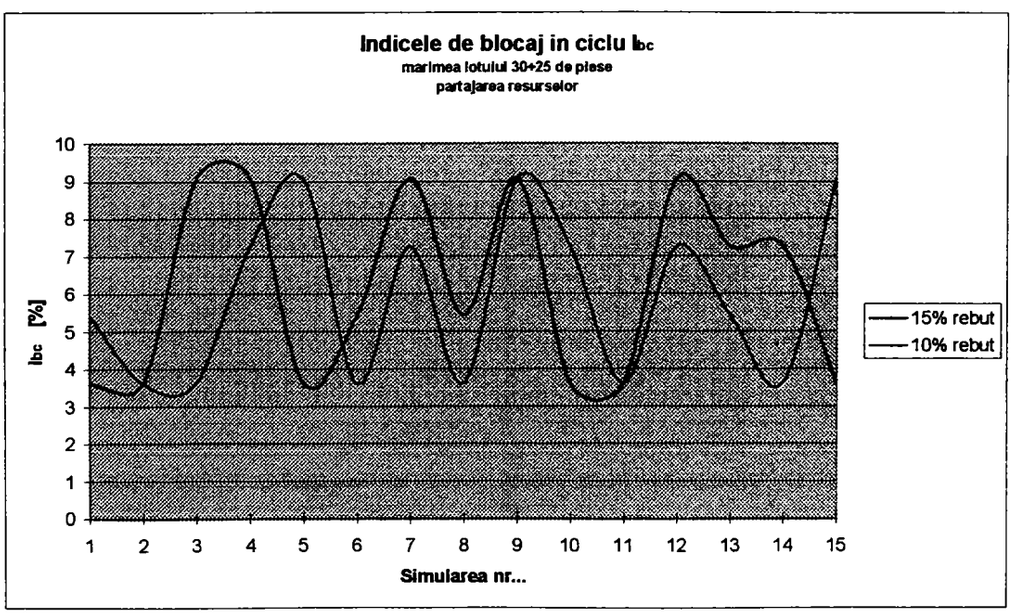




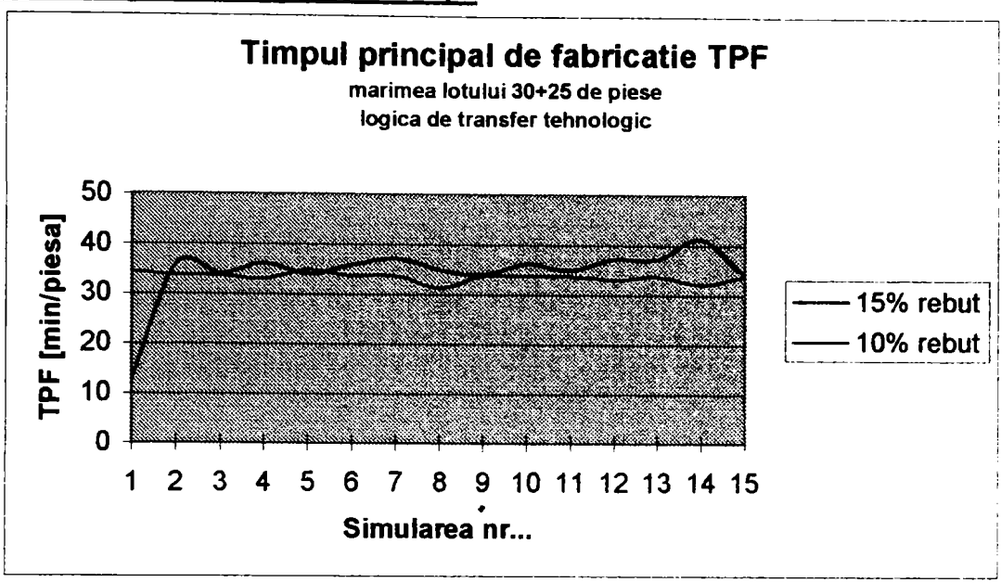


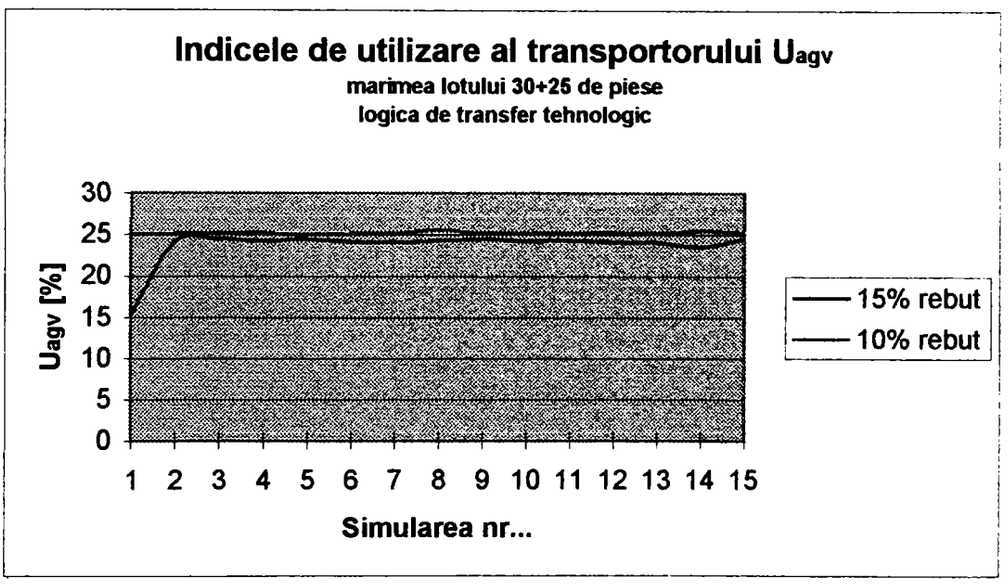

Coeficientul de utilizare al robotului de sudare Usud
marimea lotului 30+25 de piese
partajarea resurselor
Usud [%]
Simularea nr...
15% rebut
10% rebut

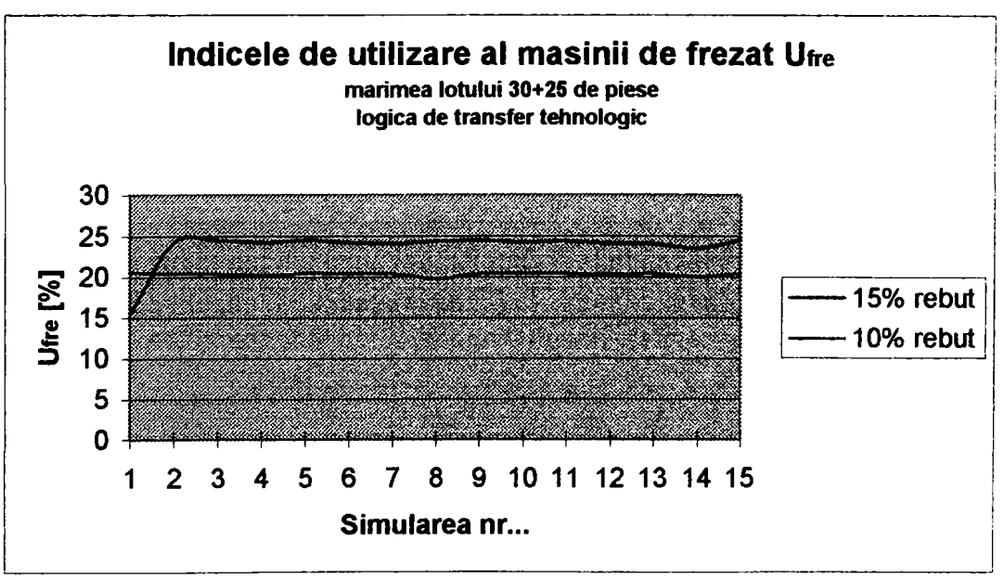

Indicele de blocaj Ib
marimea lotului 30+25 de piese
partajarea resurselor
Ib [%]
Simularea nr...
15% rebut
10% rebut

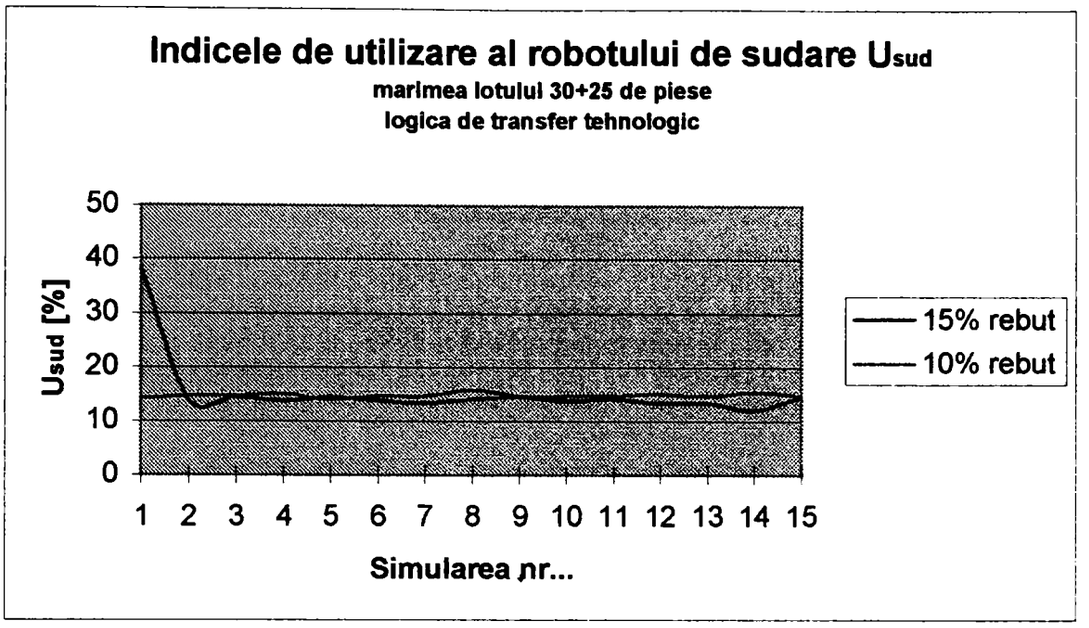

Indicele de blocaj in ciclu Ibc
marimea lotului 30+25 de piese
partajarea resurselor
Ibc [%]
Simularea nr...
15% rebut
10% rebut

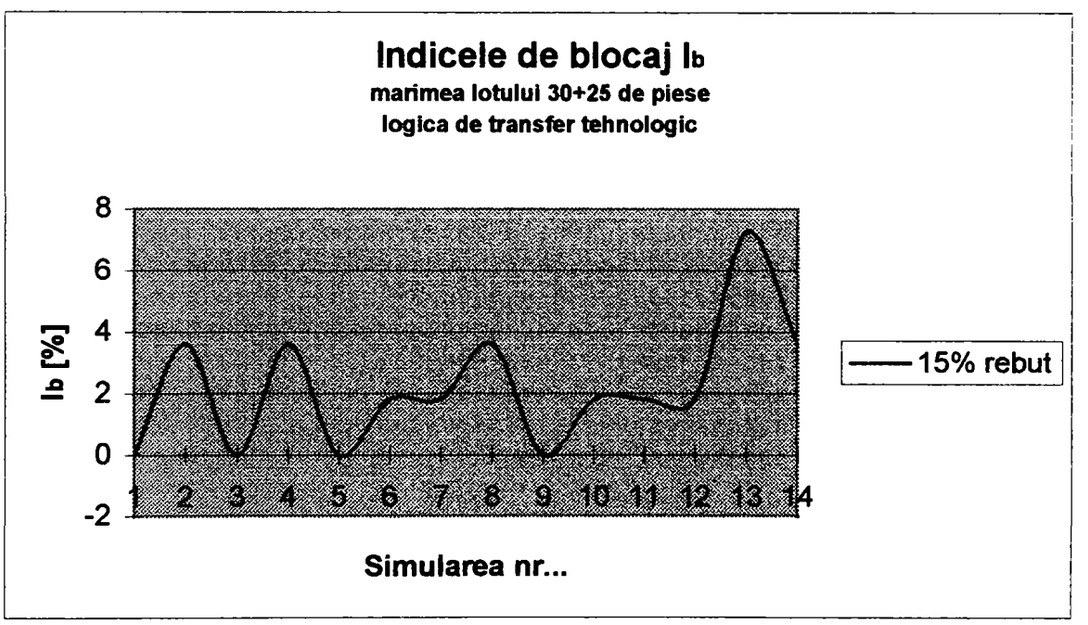
Anexa 3-4 Diagramele de variaţie a indicilor de performanţă pentru *lotul 30+25* şi *logica de transfer bazată pe tehnologia de execuţie*

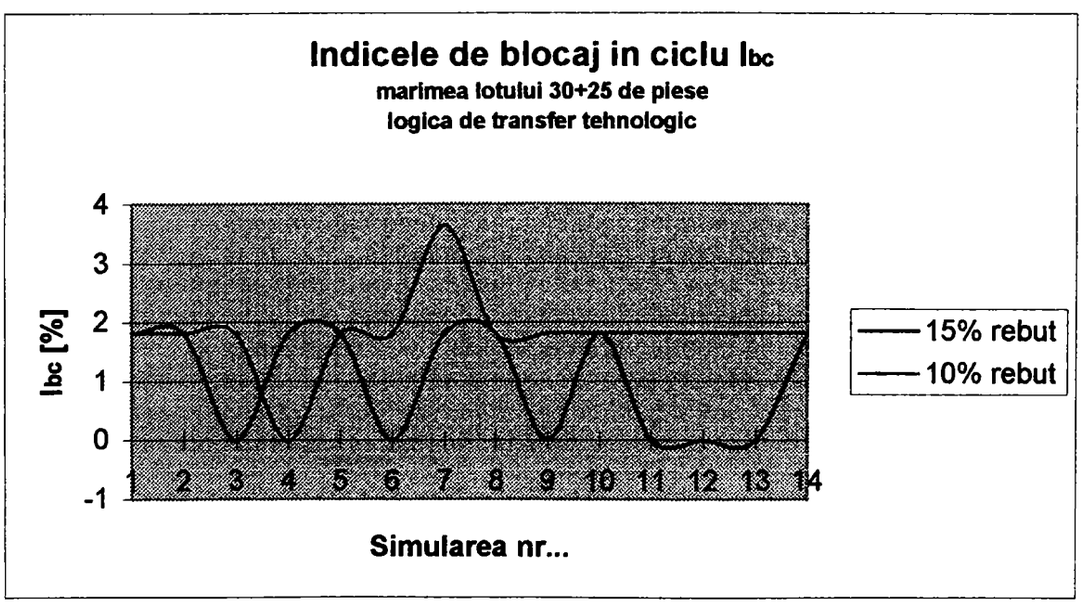




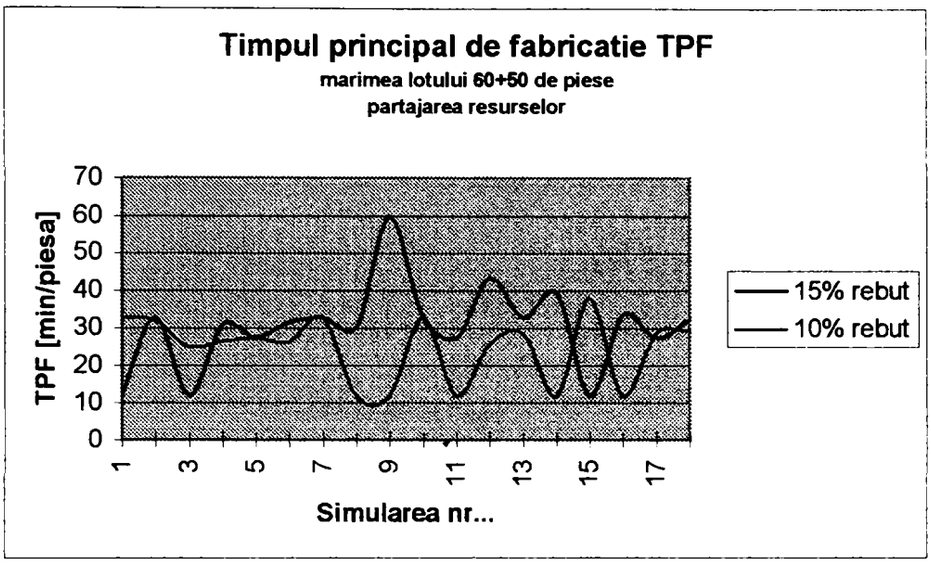


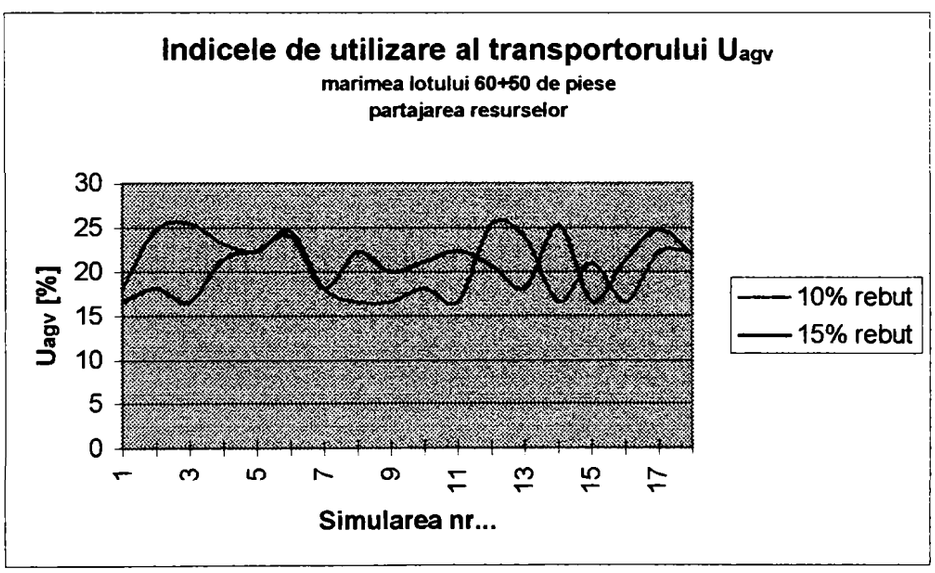

Indicele de utilizare al robotului de sudare Usud
marimea lotului 30+25 de piese
logica de transfer tehnologic
Usud [%]
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Simularea nr...
15% rebut
10% rebut

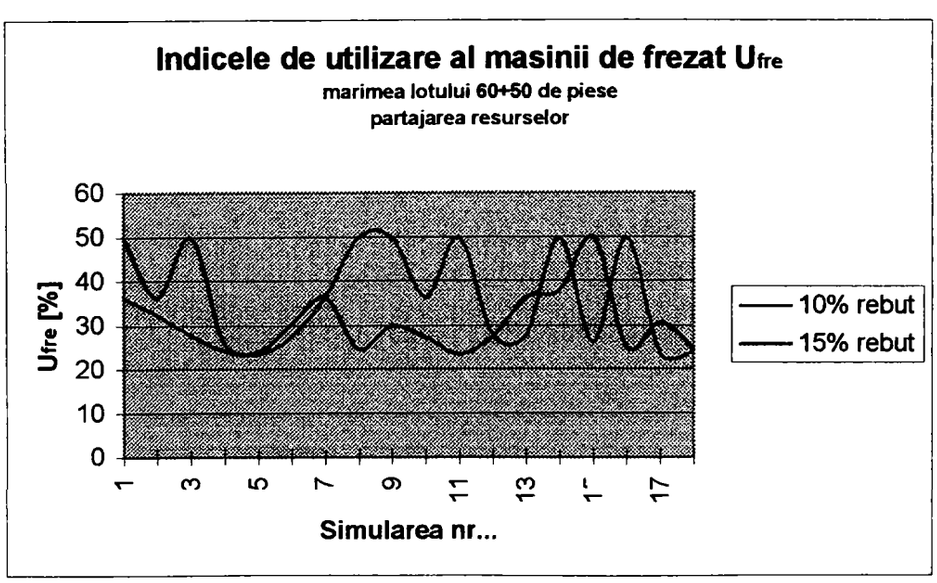

Indicele de blocaj Ib
marimea lotului 30+25 de piese
logica de transfer tehnologic
Ib [%]
8
6
4
2
0
-2
1 2 3 4 5 6 7 8 9 10 11 12 13 14
Simularea nr...
15% rebut

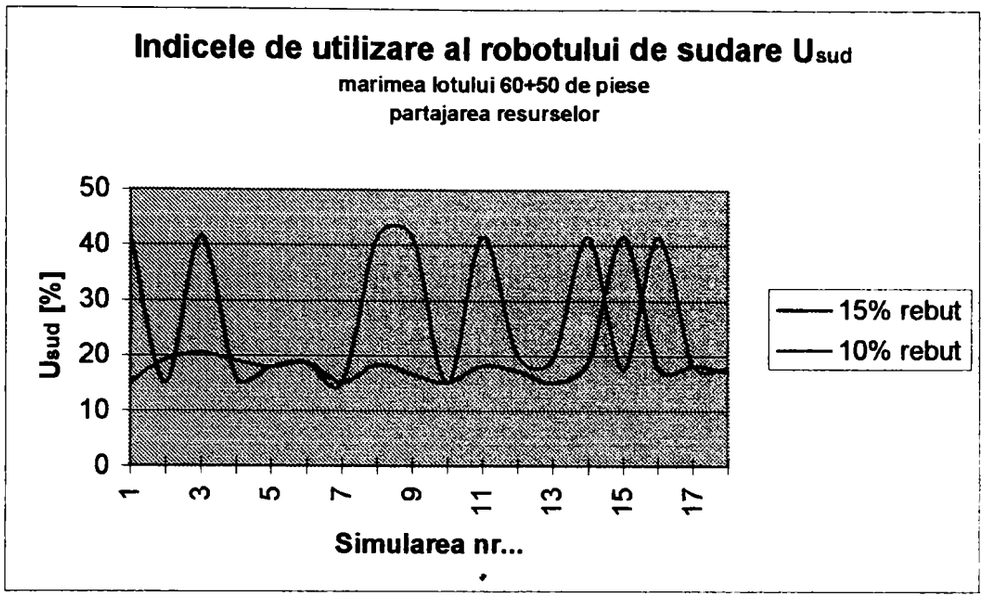

Indicele de blocaj in ciclu Ibc
marimea lotului 30+25 de piese
logica de transfer tehnologic
Ibc [%]
4
3
2
1
0
-1
1 2 3 4 5 6 7 8 9 10 11 12 13 14
Simularea nr...
15% rebut
10% rebut

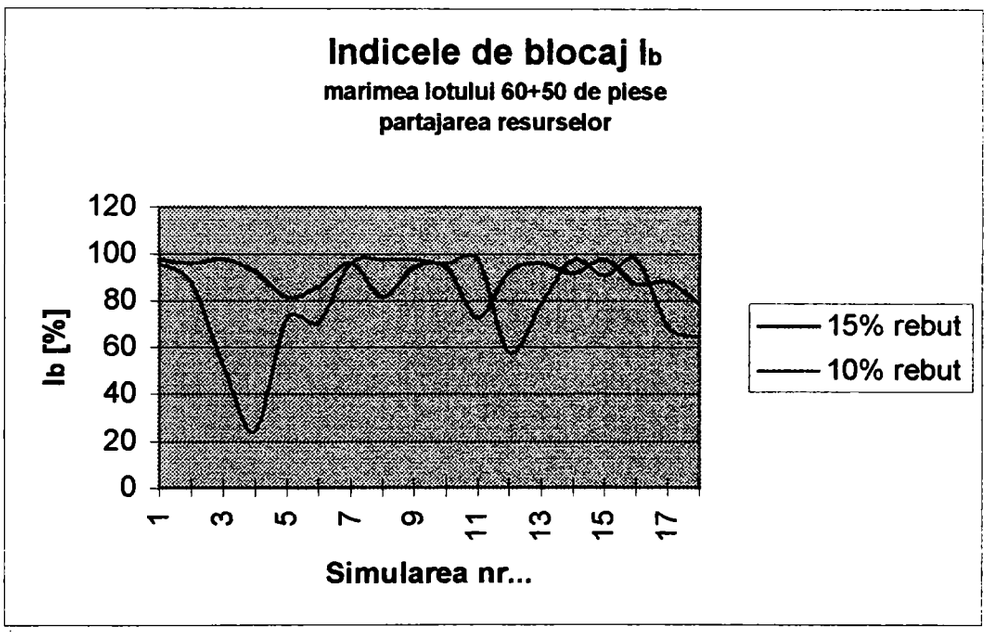
Anexa 3-5 Diagramele de variație a indicilor de performanță pentru *lotul 60+50* și *logica de transfer bazată pe partajarea resurselor*

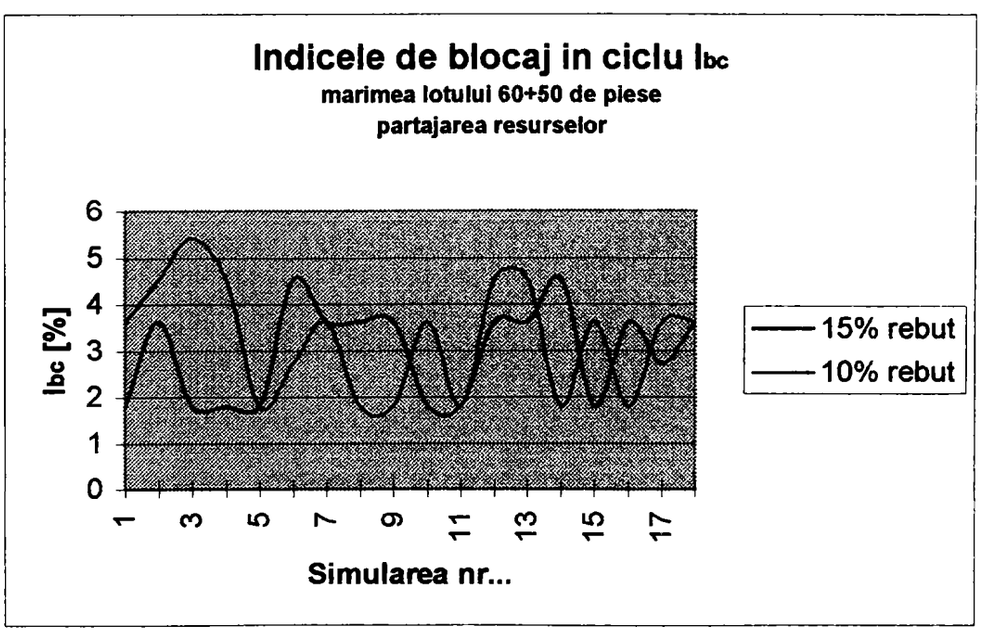




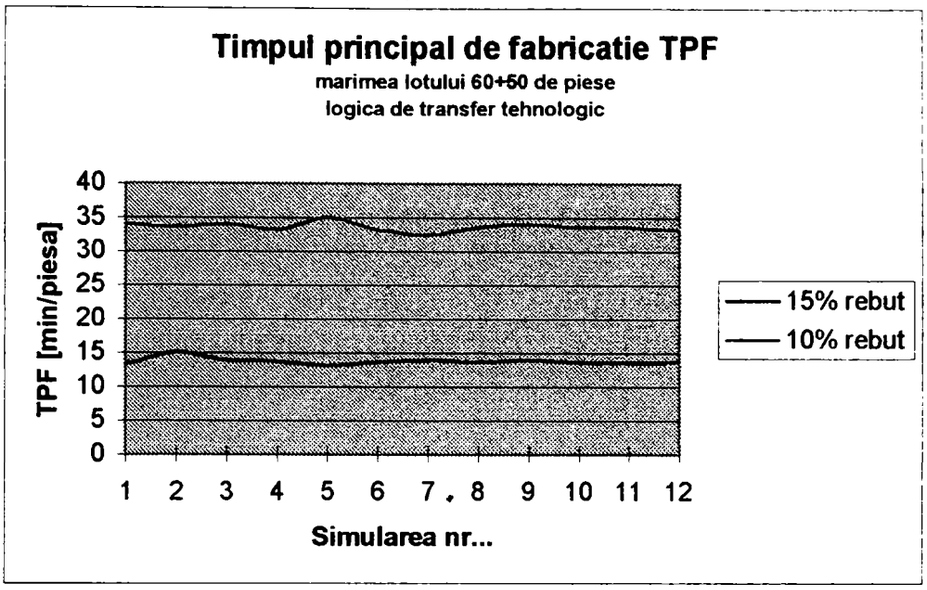


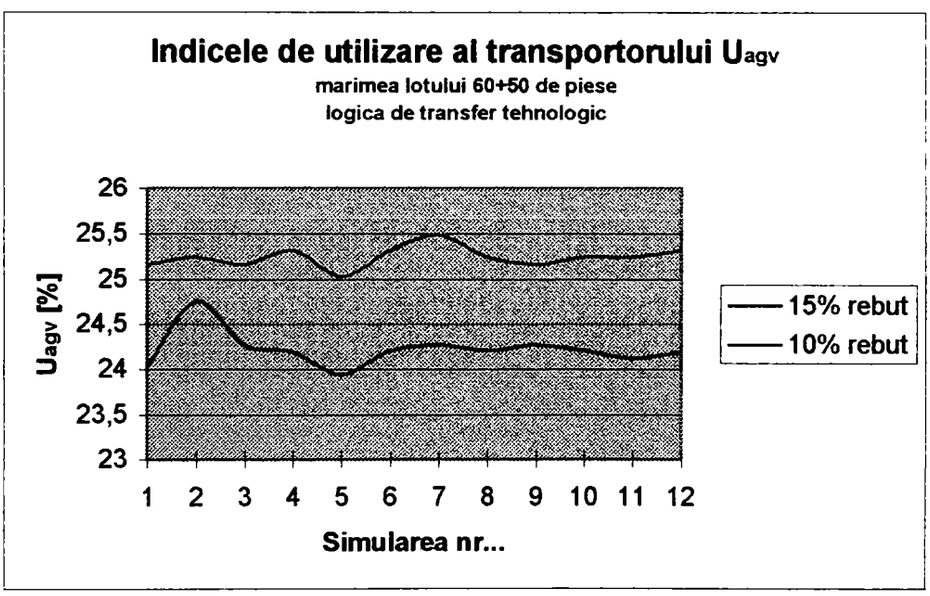

Indicele de utilizare al robotului de sudare Usud
marimea lotului 60+50 de piese
partajarea resurselor
Usud [%]
50
40
30
20
10
0
1
3
5
7
9
11
13
15
17
Simularea nr...
15% rebut
10% rebut

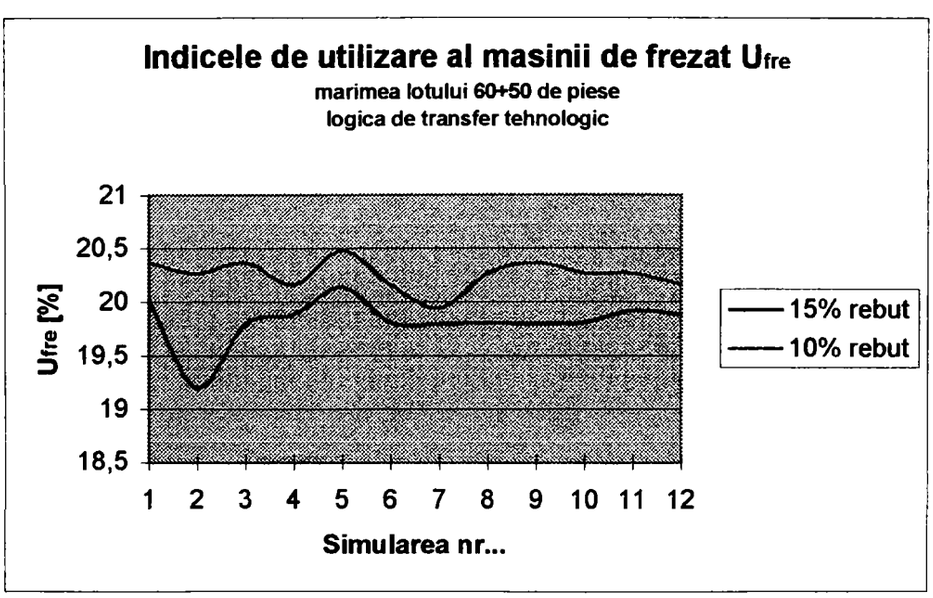

Indicele de blocaj Ib
marimea lotului 60+50 de piese
partajarea resurselor
Ib [%]
120
100
80
60
40
20
0
1
3
5
7
9
11
13
15
17
Simularea nr...
15% rebut
10% rebut

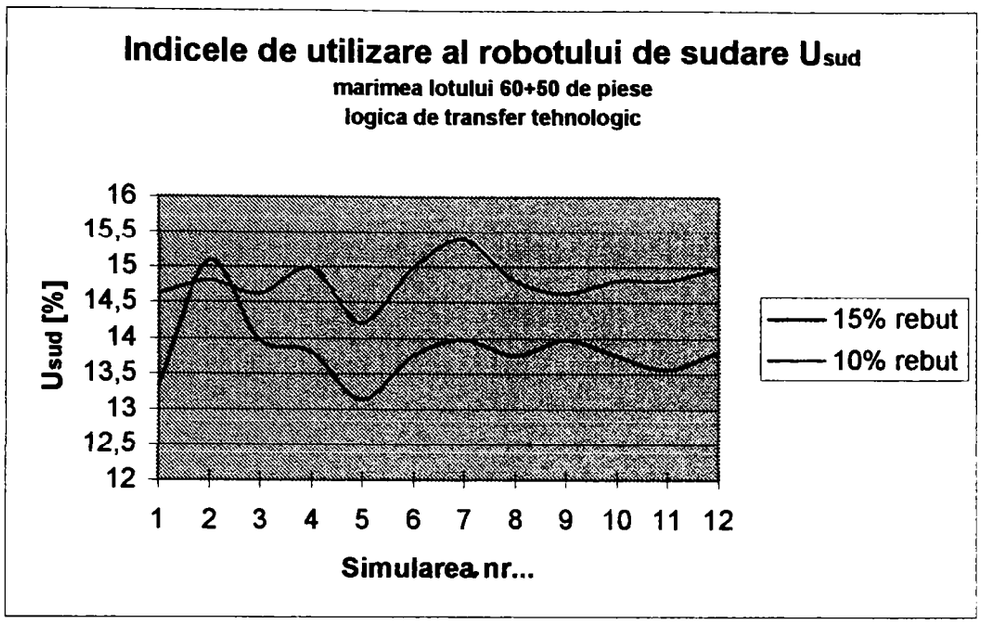

Indicele de blocaj in ciclu Ibc
marimea lotului 60+50 de piese
partajarea resurselor
Ibc [%]
6
5
4
3
2
1
0
1
3
5
7
9
11
13
15
17
Simularea nr...
15% rebut
10% rebut

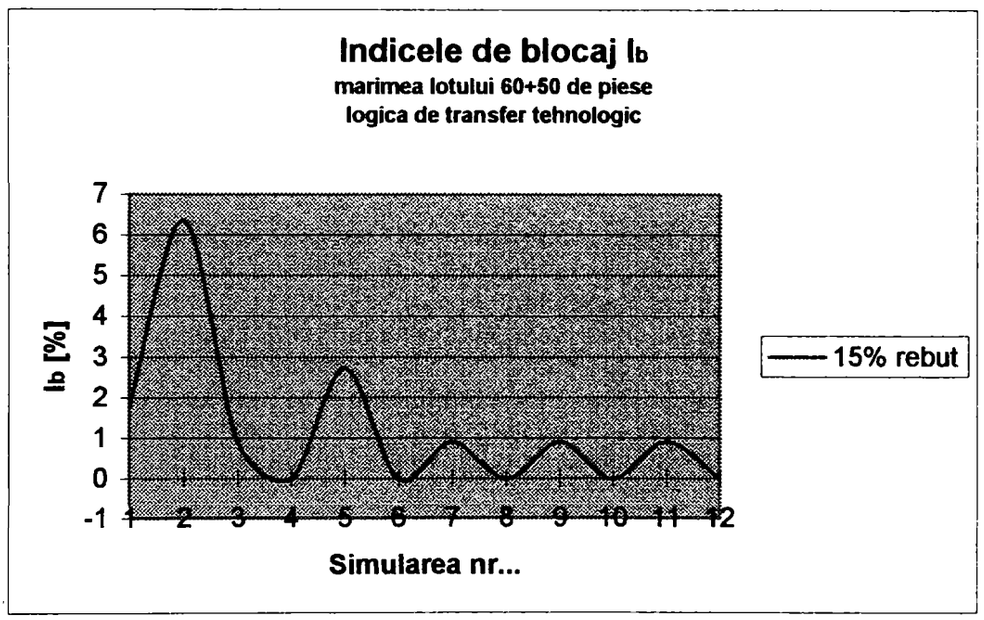
Anexa 3-6 Diagramele de variaţie a indicilor de performanţă pentru *lotul 60+50* şi *logica de transfer bazată pe tehnologia de execuţie*

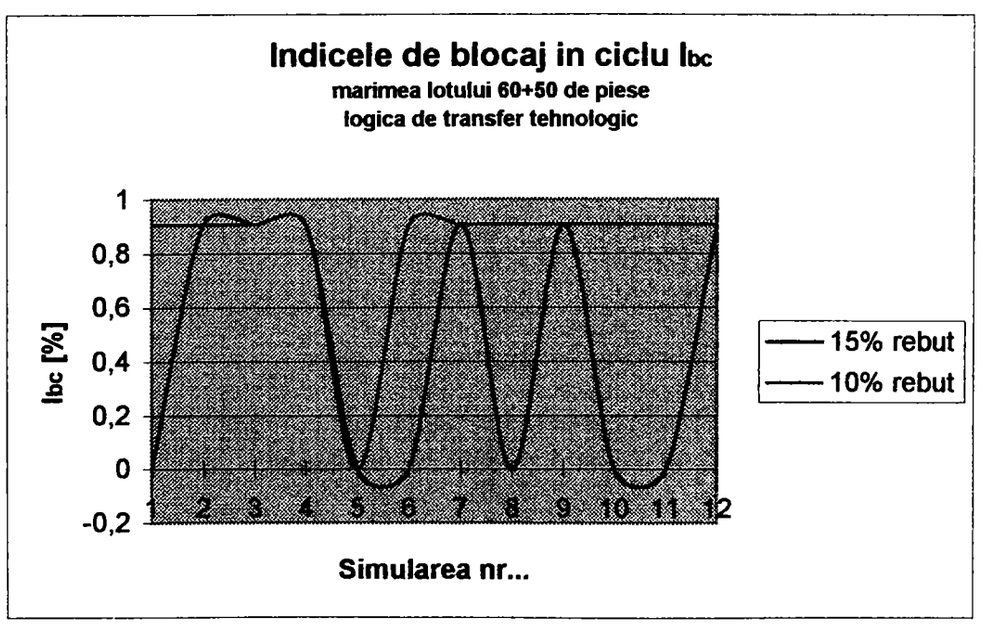




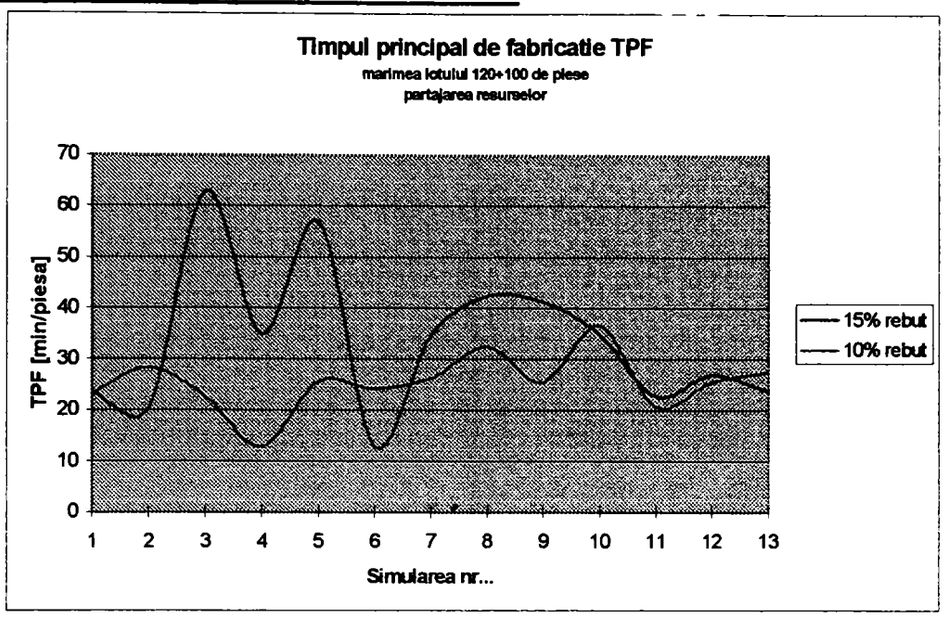


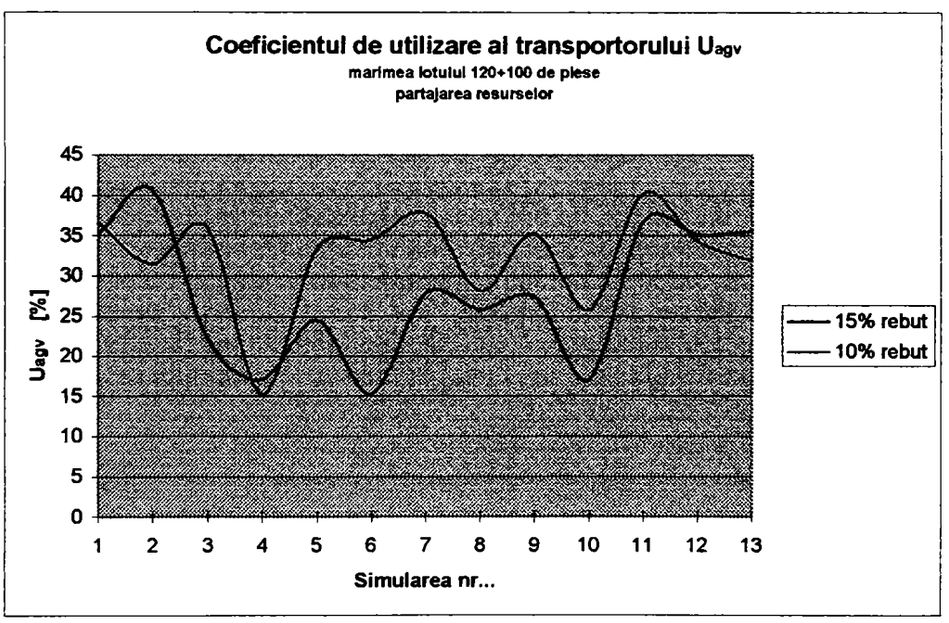

Indicele de utilizare al robotului de sudare Usud
marimea lotului 60+50 de piese
logica de transfer tehnologic
Usud [%]
16
15,5
15
14,5
14
13,5
13
12,5
12
1 2 3 4 5 6 7 8 9 10 11 12
Simularea nr...
15% rebut
10% rebut

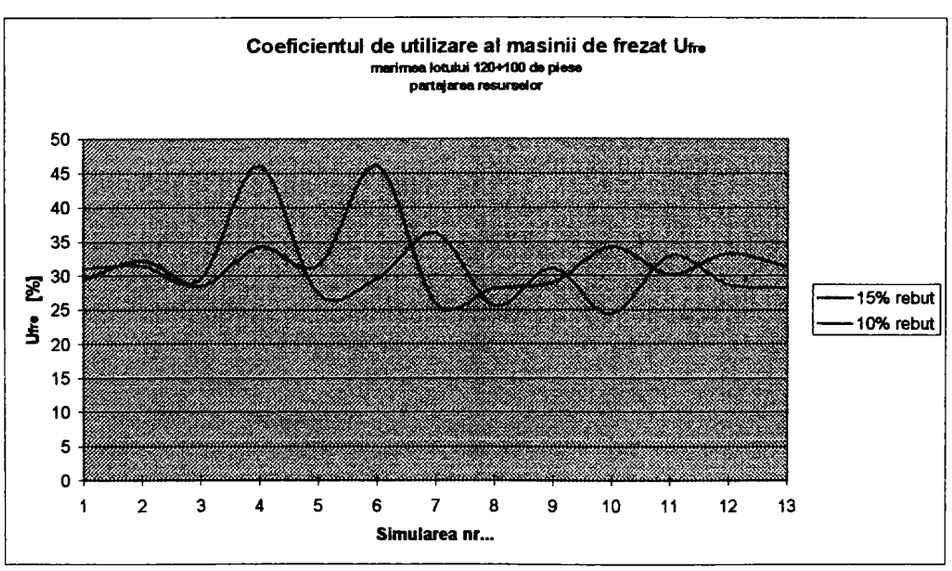

Indicele de blocaj Ib
marimea lotului 60+50 de piese
logica de transfer tehnologic
Ib [%]
7
6
5
4
3
2
1
0
-1
1 2 3 4 5 6 7 8 9 10 11 12
Simularea nr...
15% rebut

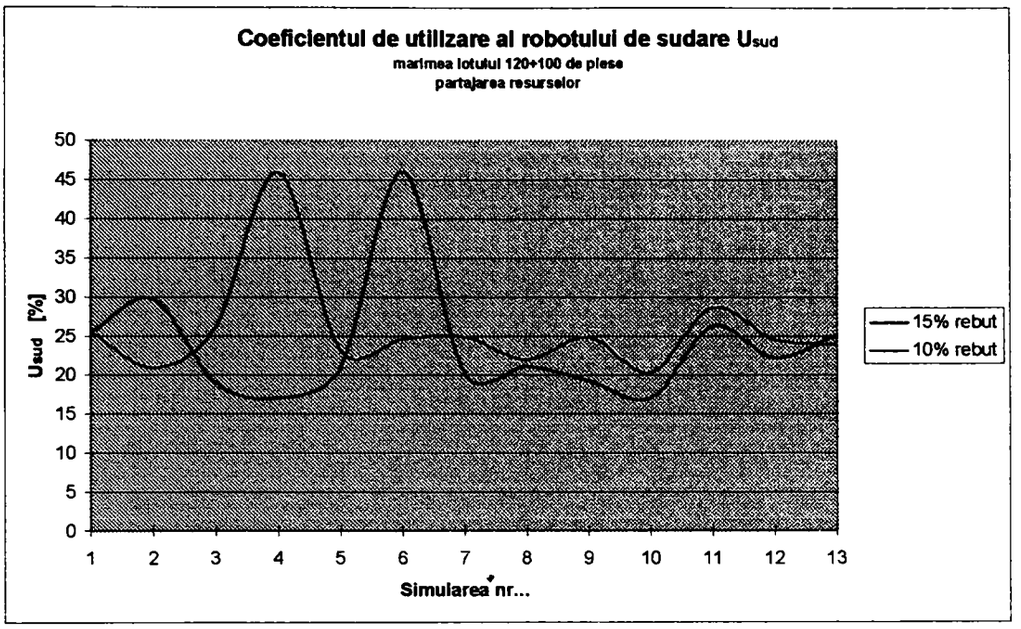

Indicele de blocaj in ciclu Ibc
marimea lotului 60+50 de piese
logica de transfer tehnologic
Ibc [%]
1
0,8
0,6
0,4
0,2
0
-0,2
1 2 3 4 5 6 7 8 9 10 11 12
Simularea nr...
15% rebut
10% rebut

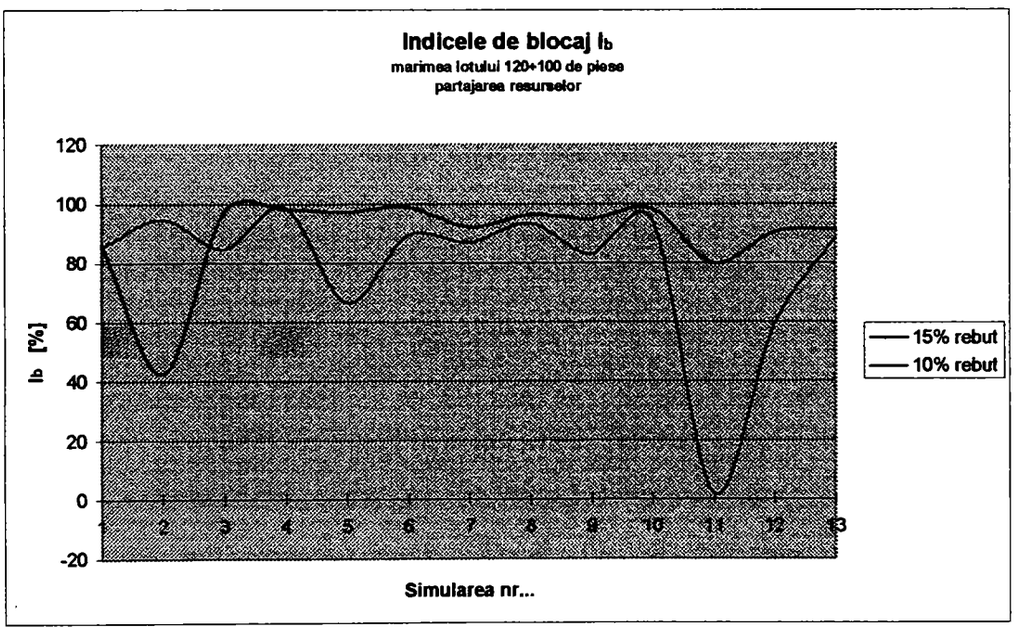
Anexa 3-7 Diagramele de variaţie a indicilor de performanţă pentru *lotul 120+100* şi *logica de transfer bazată pe partajarea resurselor*

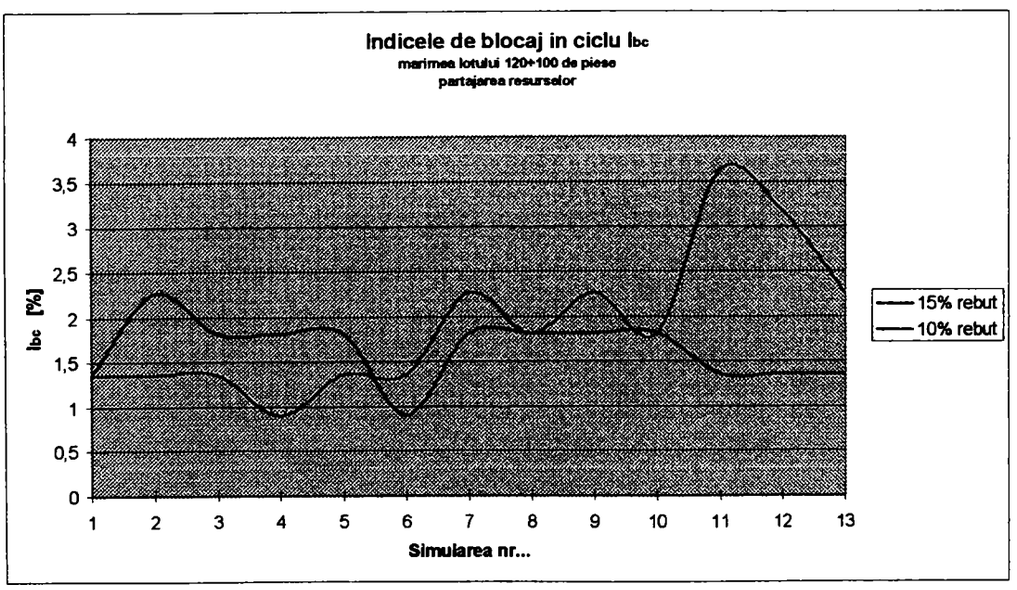




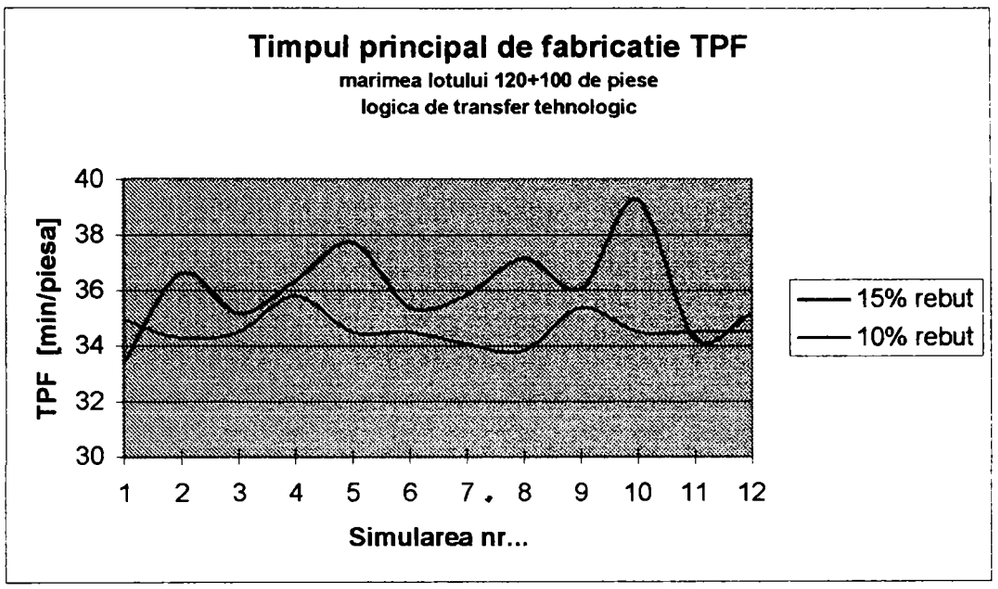


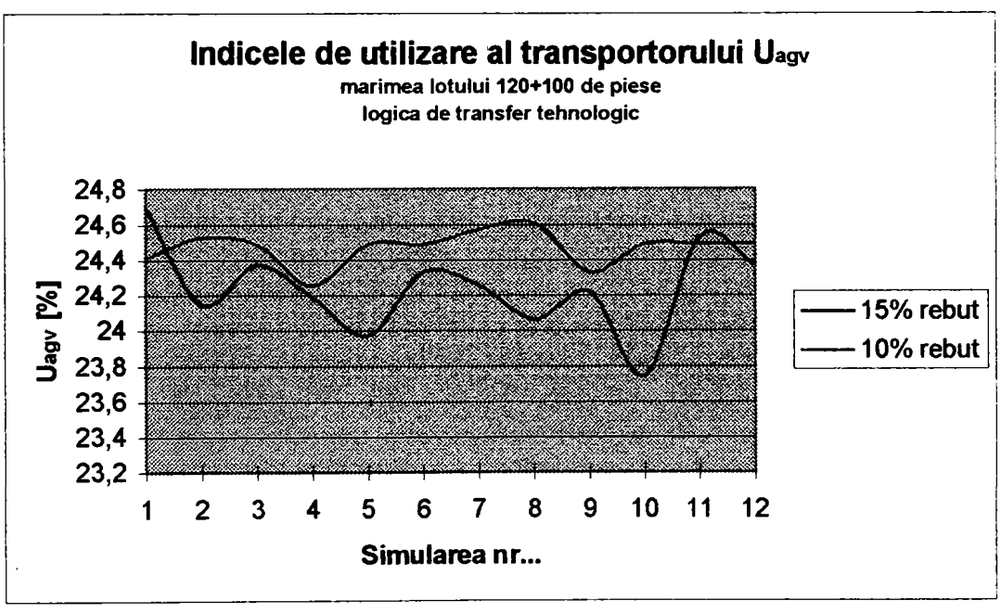

Coeficientul de utilizare al robotului de sudare Usud
marimea lotului 120+100 de piese
partajarea resurselor
Usud [%]
Simularea nr...
15% rebut
10% rebut

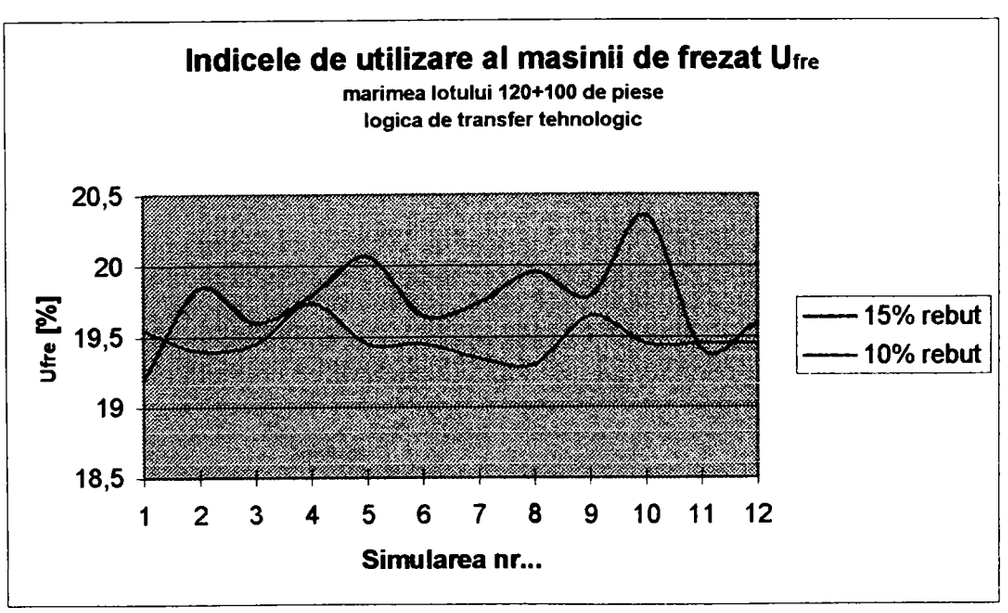

Indicele de blocaj Ib
marimea lotului 120+100 de piese
partajarea resurselor
Ib [%]
Simularea nr...
15% rebut
10% rebut

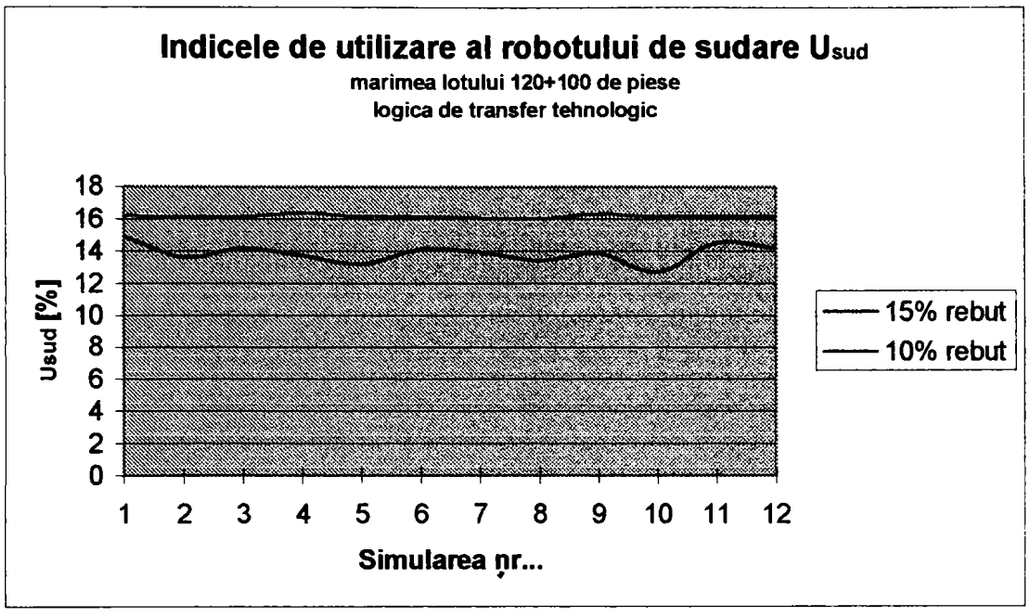

Indicele de blocaj in ciclu Ibc
marimea lotului 120+100 de piese
partajarea resurselor
Ibc [%]
Simularea nr...
15% rebut
10% rebut

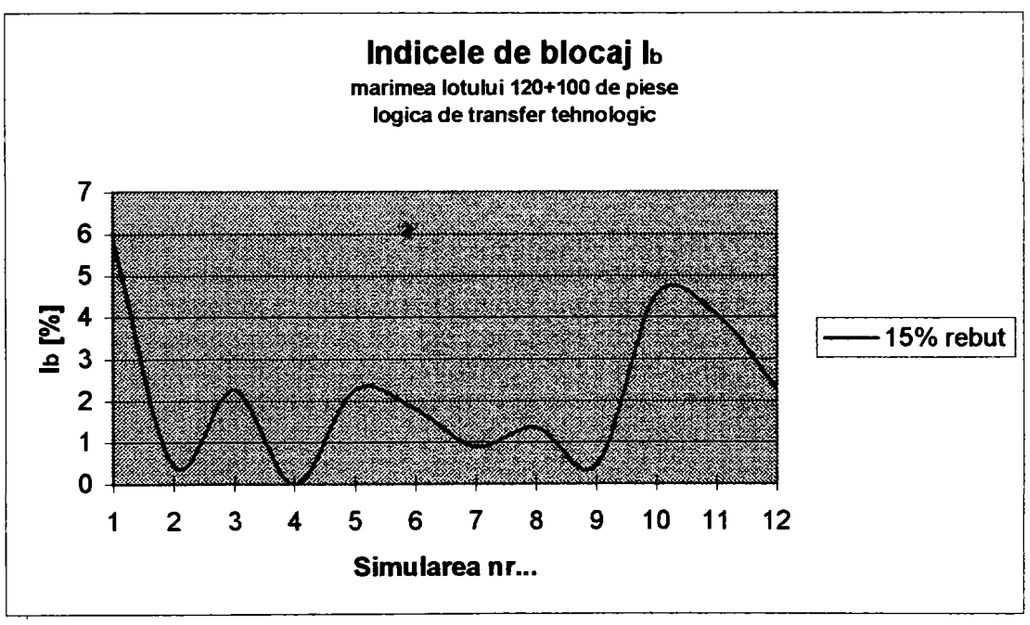
Anexa 3-8 Diagramele de variaţie a indicilor de performanţă pentru *lotul 120+100* şi *logica de transfer bazată pe tehnologia de execuţie*

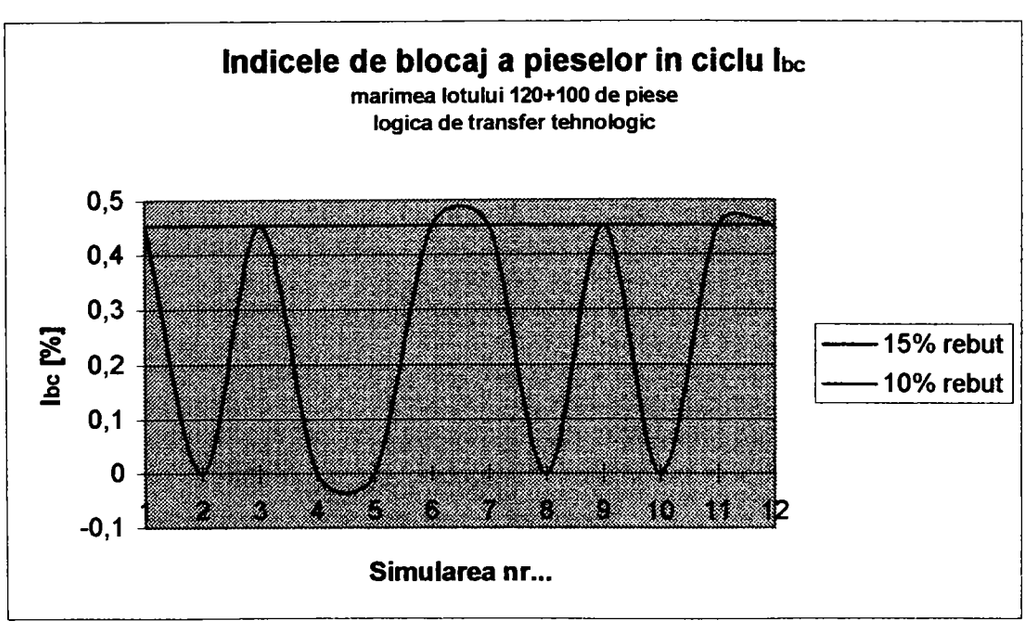







Indicele de utilizare al robotului de sudare Usud
marimea lotului 120+100 de piese
logica de transfer tehnologic
Usud [%]
18
16
14
12
10
8
6
4
2
0
1 2 3 4 5 6 7 8 9 10 11 12
Simularea nr...
15% rebut
10% rebut


Indicele de blocaj Ib
marimea lotului 120+100 de piese
logica de transfer tehnologic
Ib [%]
7
6
5
4
3
2
1
0
1 2 3 4 5 6 7 8 9 10 11 12
Simularea nr...
15% rebut


Indicele de blocaj a pieselor in ciclu Ibc
marimea lotului 120+100 de piese
logica de transfer tehnologic
Ibc [%]
0,5
0,4
0,3
0,2
0,1
0
-0,1
1 2 3 4 5 6 7 8 9 10 11 12
Simularea nr...
15% rebut
10% rebut

3.7. Bibliografie

1. ***, 1986, Japan - USA Symposium on Flexibel Automation.
2. David, R., Alla, H., 1992, Petri Nets and Grafcet. Tools for Modeling Descrete Events Systems. Prentice Hall, London, New York, Toronto.
3. Davidoviciu, A., Drăgănoiu, Gh., Moangă, A., 1986, Modelarea, simularea şi comanda manipulatoarelor şi roboţilor industriali. Editura Tehnică, Bucureşti.
4. Di Mascolo, M., Frein, Y., Dallery, Y., David, R., 1990, A Unifiedd Modeling of Kanban Systems using Petri Nets in International Journal of Flexible Manufacturing Sytems.
5. Doo Yong Lee, Frank DiCesare, 1995, Petri Net-Based Heuristic Scheduling for Flexible Manufacturing. In "Petri Nets in Flexible and Agile Automation", Kluewer Academic Publishers, Boston , Dordrecht, London.
6. Dreucean, M., 1996, Modeling and Simulation-Tools for Evaluation in Flexible Manufacturing Systems, Institute of Technology, Kuopio, Finland.
7. Dreucean, M., 1996, Stadiul actual al cercetărilor şi realizărilor în domeniul sistemelor de fabricaţie flexibilă robotizate, Referatul nr. 1, Timişoara.
8. Dreucean, M., 1997, Modelarea fluxurilor de materiale şi de informaţie în sisteme de fabricaţie flexibile, Referatul nr. 2, Timişoara.
9. Dreucean, M., Ioanovici, Fr., jr., 1996, Maşini de lucru în procese robotizate. Editura "Politehnica", Timişoara.
10. Jensen, K., 1992, Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts. Springer-Verlag, Berlin, London, New-York, Toronto.
11. Kovacs, Fr., Gheorghiu, N., Dreucean, M., 1989, Asupra posibilităţilor de robotizare a operaţiilor de nituire în Industria aeronautică. Simpozion "Robot", Baia Mare, 1989.
12. Kovacs, Fr., Grigorescu, S., Rădulescu, C., 1994, Sisteme de fabricaţie flexibilă robotizate, Părţile I şi II, Litografia UPT, Timişoara.
13. Picoş, C., Ailincăi, Gh., Bohosievici, C., Pruteanu, O., Coman, Gh., Braha, V., Paraschiv, D., 1974, Calculul adaosurilor de prelucrare şi al regimurilor de aşchiere. Editura Tehnică, Bucureşti.
14. Picoş, C., Coman, Gh., Dobre, N., Pruteanu, O., Rusu, C., Rusu, St., trufinescu, St., 1982, Normarea tehnică pentru prelucrări prin aşchiere. Editura Tehnică, Bucureşti.
15. Ralston, A., 1983, Encyclopedia of Computer Science and Engineering, Second Edition. Van Nostrand Reinhold Company, New York.
16. Shifang Li, Toshi Takamori, Satoshi Tadokoro, 1995, Scheduling and Rescheduling of AGVs for Flexible and Agile Manufacturing. In "Petri Nets in Flexible and Agile Automation", Kluewer Academic Publishers, Boston , Dordrecht, London.
17. Silva, M., Valette, R., 1990, Petri Nets and Fexible Manufacturing. Advances in PN'89, LNCS 84, Springer-Verlag, Berlin.
18. Takashi Sato, Kazuo Nose, 1995, Automatic Generation of Sequence Control Programs via Petri Nets and logic Tables for Industrial Applications. In "Petri Nets in Flexible and Agile Automation", Kluewer Academic Publishers, Boston , Dordrecht, London.
19. Teicholz, E., Joel, n. Orr, 1987, Computer Integrated Manufacturing Handbook. McGraw-Hill Book Company, New-York.
20. Viswanadham, N., Narahari, Y., 1992, Performance Modeling of Automated Manufacturing. (Prentice-Hall, Englewood Cliffs, New Jersey).
21. Vlase, A., Sturzu, A., Neagu, C., Stanescu, C., Atanase, M., 1989. Tehnologii de prelucrare pe strunguri. Editura Tehnică, Bucureşti.
22. Widman, L., Lopato, A., Niel, R., 1989, Artificial Inteligence, Simulation and Modeling. John Wiley & Sons, New York, Chichester, Brisbane, Toronto, Singapore.

4. PROBLEME DE OPTIMIZARE A FLUXURILOR ÎN SISTEME FLEXIBILE DE FABRICAȚIE

4.1. Cuprins

In capitolele anterioare au fost prezentate diverse particularități funcționale ale sistemelor de fabricație flexibilă. Așa după cum este prezentat în capitolul 1 §1.7, una din problemele de maximă importanță este *identificarea condițiilor de apariție a blocajelor și prevenirea apariției acestora*. În capitolul de față sunt prezentate încercările teoretice ale autorului îndreptate în această direcție, precum și descrierea aplicației practice prin care s-a pus în evidență va;abilitatea soluției propuse. Programele elaborate se bazează pe datele obținute în urma simulării funcționării sistemului cu rețele Petri iar *algoritmul de conducere a sistemului în vederea evitării blocajelor* precum și programele "C" scrise în acest scop sunt originale.

La baza acestor încercări de conducere a sistemului pentru evitarea blocajelor stau observațiile legate de cauzele care produc în general blocaje într-un sistem de fabricație așa cum au fost ele definite teoretic în capitolele anterioare și așa cum au rezultat în urma analizei cu rețele Petri din capitolul anterior. Pe baza acestor concluzii s-a trecut la formularea unei *strategii de detectare a nodurilor de blocaj* din sistem, înțelegând printr-un "nod" o anumită stare stabilă. In final, cu ajutorul unor programe scrise în C, s-a reușit conducerea sistemului astfel încât **să fie evitate nodurile de blocaj, adică acele stări ale sistemului care nu mai au nici un succesor, deci care nu oferă nici o perspectivă ca sistemul să mai funcționeze**.

4.2. Graful de incidență al nodurilor funcționale ale unui sistem de fabricație

Modelul cu rețele Petri construit pentru sistemul de fabricație prezentat în capitolul anterior permite trasarea unui *graf de incidență* corespunzător tuturor nodurilor funcționale ale sistemului. Acesta se regăsește în teoria generală a rețelelor Petri sub denumirea de "Occurence Graph". Determinarea acestui graf poate fi făcută cu ajutorul meniurilor mediului grafic în care este rulată simularea sau cu ajutorul unor secvențe de cod asociate unor tranziții ale modelului.

Fiecare nod din graful astfel obținut va reprezenta o anumită stare a sistemului. Prin afișarea *descriptorului de nod* se poate vedea conținutul real de jetoane din fiecare poziție din model. In Figura 4.2-1 este redat un astfel de fișier text corespunzător nodului 240 al grafului de incidență. Comparând acest fișier cu diagramele prezentate în anexele capitolului anterior se observă că apar toate pozițiile din paginile diagramei, fără excepție. De asemenea se poate observa că toate nodurile au un descriptor care respectă aceeași structură, lucru important pentru procedeele de analiză ulterioară a acestora.

Trecerea de la un nod la altul în evoluția reală a sistemului este dirijată de arcele de legătură dintre noduri. Fiecare arc asigură legătura dintre un nod de origine și unul dintre succesorii lui, astfel încât rețeaua de noduri care formează graful de incidență se dezvoltă din aproape în aproape până la epuizarea tuturor stărilor posibile. Unele dintre noduri nu mai au însă nici un succesor, astfel încât sistemul odată ajuns în această stare nu mai are nici o posibilitate de dezvoltare ulterioară. Se spune că s-a ajuns în acest moment într-o *stare de blocaj* a cărei cauză poate fi oricare dintre cele prezentate în capitolul 1. Un fragment din graful de incidență al modelului utilizat este redat în Figura 4.2-2 Trecerea de la un nod la altul se face prin declanșarea unor tranziții asociate fiecărui arc al diagramei. Declanșarea unei tranziții va schimba starea sistemului prin modificarea conținutului de jetoane ale unora dintre pozițiile din diagramă. Se ajunge astfel la o nouă stare, care satisface condițiile de declanșare ale altor tranziții și așa mai departe. Descriptorul asociat unui arc conține toate informațiile referitoare la tranziția a cărei

```
240
depozit1'Depozit1 1: 2`("Ansamblu",0,0,0,0)@[49,59]
depozit1'Robot1 1: 1`disp@[69]
depozit1'CntrR1 1: 1`10
depozit1'CtrlPs 1: 1`Frezare
depozit1'AGV 1: 1`("",0,0,0,0)@[69]
depozit2'AGV 1: 1`("",0,0,0,0)@[69]
depozit2'Dep2 1: 1`("Rebut",0,0,0,0)@[69]
depozit2'Robot1 1: 1`disp@[69]
depozit2'CntrR1 1: 1`10
depozit2'CtrlPs 1: 1`Frezare
conveior'Conv 1: 1`("Flansa2",0,0,0,4)@[0]
conveior'AGV 1: 1`("",0,0,0,0)@[69]
conveior'Robot1 1: 1`disp@[69]
conveior'CntrR1 1: 1`10
conveior'CtrlPs 1: 1`Frezare
sudare'AGV 1: 1`("",0,0,0,0)@[69]
sudare'InCloos 1: tempty
sudare'OpUman 1: 1`disp@[56]
sudare'OutCloos 1: tempty
sudare'AGV1 1: 1`("",0,0,0,0)@[69]
sudare'CntrOp 1: 1`5
sudare'CtrlPs 1: 1`Frezare
frezare'InMF 1: tempty
frezare'Robot2 1: 1`disp@[72]
frezare'outMF 1: tempty
frezare'InMas 1: 1`("Flansa1",0,2,3,4)@[72]
frezare'CntrR2 1: 1`13
frezare'CntrFr 1: 1`5
frezare'Control 1: 1`Frezare
frezare'AGV 1: 1`("",0,0,0,0)@[69]
masurare'Robot2 1: 1`disp@[72]
masurare'OutMas 1: tempty
masurare'AGV 1: 1`("",0,0,0,0)@[69]
masurare'Robot3 1: 1`disp@[67]
masurare'InAsmb 1: 1`("Flansa1",0,0,3,4)@[67]
masurare'InMas 1: 1`("Flansa1",0,2,3,4)@[72]
masurare'CntrR2 1: 1`13
masurare'NrMas 1: 1`4
masurare'CtrlPs 1: 1`Frezare
asamblare'Robot2 1: 1`disp@[72]
asamblare'AGV 1: 1`("",0,0,0,0)@[69]
asamblare'InAsmb 1: 1`("Flansa1",0,0,3,4)@[67]
asamblare'CntrR2 1: 1`13
asamblare'CtrlPs 1: 1`Frezare
```

Figura 4.2-1

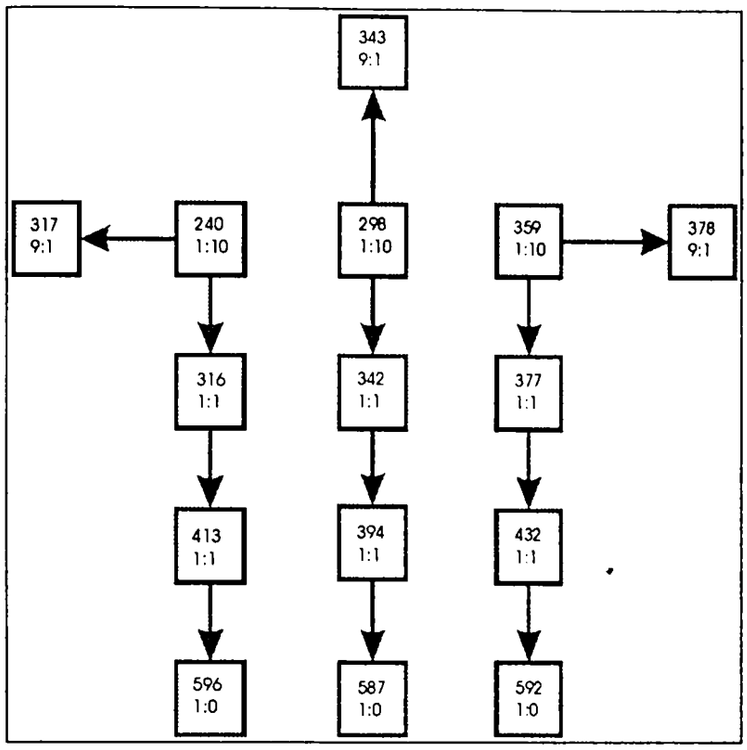


Figura 4.2-2

declanşare va determina modificarea stării sistemului dintr-un nod în altul. Figura 4.2-3 redă un astfel de descriptor.

Se poate observa în figura menționată mai sus faptul că pe prima linie a descriptorului apare numărul arcului precum şi numărul nodului de start şi a celui de sfârşit. Tranziția 748 a cărei declanşare va determina schimbarea de stare a sistemului din nodul 378 în nodul 433 are denumirea "TriAGV" şi va modifica "conținutul" de jetoane al pozițiilor din pagina "conveior" a diagramei. Condițiile de validare ale tranziției sunt prezentate între cele două acolade. Prin compararea descriptorilor de nod pentru nodul de intrare şi cel de ieşire se pot determina modificările care se produc în distribuția jetoanelor din sistem în urma declanşării tranziției.

```
748:378->433
conveior'TriAGV 1:
{piesa="Flansa2",nr4=4,nr3=0,nr2=0,nr1=0,i=11,contr=1`Masurare,R1=disp}
```

Figura 4.2-3

Lucrul cel mai important legat de aceste grafuri de incidență îl constituie faptul că materializează o anumită logică de rutare, implementată la nivelul diagramei modelului prin intermediul inscripțiilor de pe arce şi al condițiilor de gardă ale tranzițiilor. Se poate astfel urmări foarte uşor efectul pe care îl are modificarea unor condiții logice asupra dezvoltării rețelei de noduri sau asupra densității nodurilor de blocaj. Coroborând concluziile extrase din analiza grafului de incidență cu aspectul rețelei şi condițiile logice impuse, se poate ajunge la o configurație a rețelei din ce în ce mai bună, mai apropiată de un optim al traseelor de rutare.

Pentru a sintetiza observațiile făcute în legătură cu aspectul unui graf de incidență şi cu informațiile foarte detaliate pe care acesta le pune la dispoziția utilizatorilor, în Figura 4.2-1 a fost redată o pagină completă a modelului utilizat în care a fost dezvoltată o structură parțială a grafului de incidență. Se poate observa şi un descriptor de nod şi unul de arc.

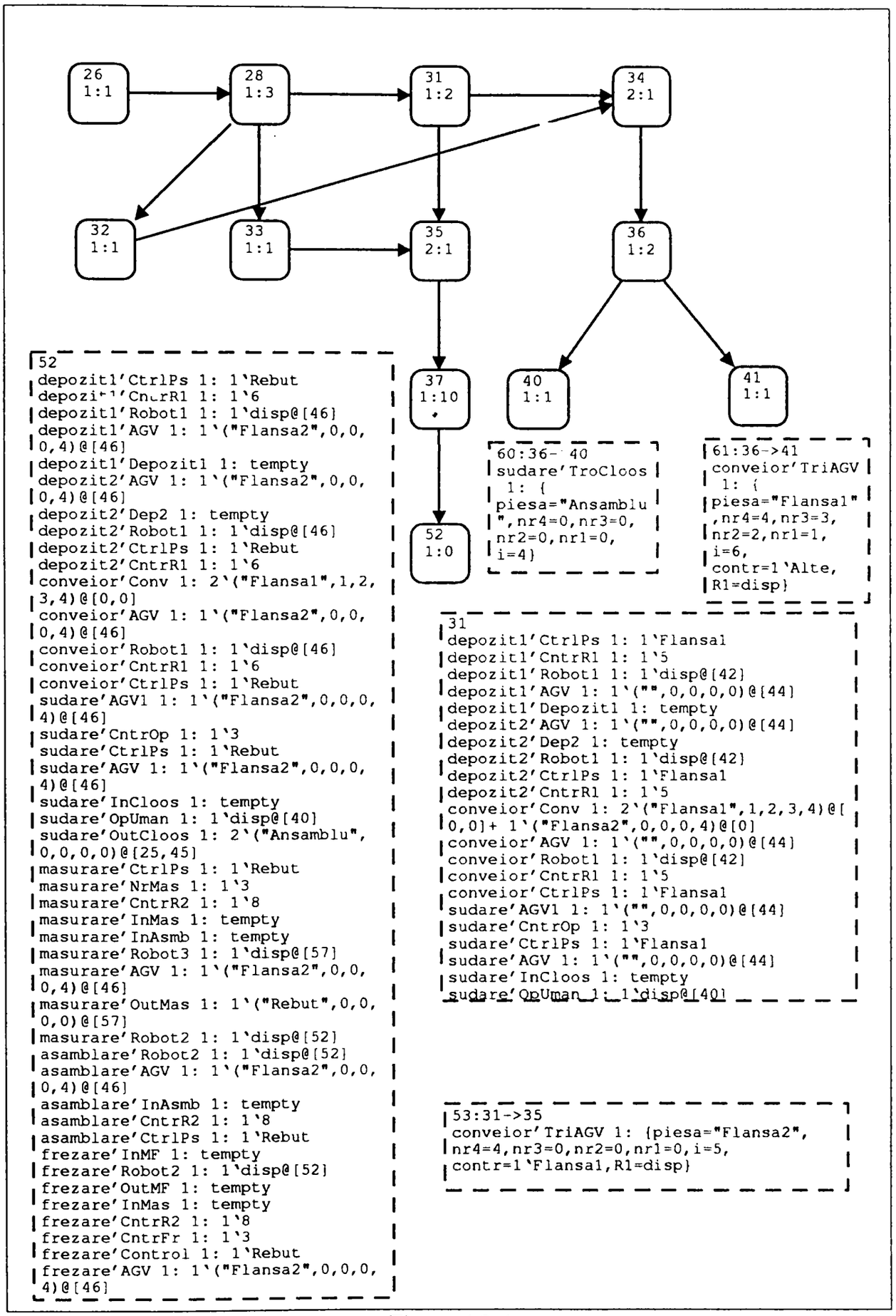


Figura 4.2-4

Graful de incidență furnizează foarte multe informații despre starea sistemului la un moment dat cât şi despre o serie de particularități de comportament ale acestuia în funcționare. De exemplu se poate observa că nodul "52" este un nod de blocaj pentru că el nu mai are nici un succesor. Inscripția de pe acest nod este "1:0" ceea ce înseamnă că el are un predecesor şi nici un succesor. Din medalionul care conține descriptorul de nod asociat nodului "52" poate fi identificată şi cauza blocajului. Astfel, se poate observa că pe una din liniile corespunzătoare paginii "măsurare" apare o poziție "CtrlPs" (denumirea prescurtată provine de la "control piesă") al cărei conținut este "1`Rebut". In mod normal această piesă ar trebui să fie transferată în depozitul "D2" de către sistemul de transfer AGV. Dacă se verifică însă conținutul dispozitivului de transfer, (poziția "AGV" de pe oricare pagină), se observă că acesta este ocupat cu o piesă de tip "Flansă2", deci el nu poate prelua piesa rebut pentru transferul ei în depozit. De asemenea nu există nici o altă variantă de transfer a piesei "Flansa2" pentru că aceasta ar trebui să se asambleze pe dispozitivul de transfer cu o piesă de tip "Falnsa1". In felul acesta practic sistemul nu mai are nici o posibilitate de evoluție fiind blocat. Pentru a evita acest blocaj ar fi util ca din nodul "31" sistemul să nu mai fie lăsat să evolueze pe direcția "35-57-52" ci să fie dirijat spre nodul "34" de unde are alte posibilități de continuare care nu duc la blocaj.

4.3. Strategia de conducere a sistemului pentru evitarea blocajelor

In acest paragraf vor fi prezentate încercările autorului de a configura un sistem de conducere a sistemului de fabricație care să ducă în final la eliminarea completă a blocajelor în timpul funcționării. Aceasta constituie de fapt esența procesului de optimizare a fluxurilor de materiale. Ideea de bază a acestuia constă în identificarea "a priori" a nodurilor de blocaj pe baza grafului de incidență şi conducerea ulterioară a sistemului astfel încât să nu treacă niciodată prin ele.

4.3.a Schema bloc a modului de conducere a sistemului în scopul evitării blocajelor

Graful de incidența care stă la baza procedeului de evitare a blocajelor care va fi descris în continuare are avantajul proncipal că redă toate stările sistemului, indiferent de logica de transfer care a fost implementată la nivelul modelului. Oricare din stările posibile în care se poate afla sistemul vor apare în lista de noduri. Unele dintre noduri constituie după cum s-a văzut adevărate "macaze" pentru evoluția sistemului. Angajarea sistemului pe una din

căile de evoluție posibile în momentul atingerii unui astfel de "macaz" se face aleator în cadrul simulării. In realitate lucrurile se prezintă altfel, deoarece există întotdeuna factori obiectivi sau subiectivi care să determine o anumită opțiune din cele posibile. Totuşi, nimic nu poate anticipa angajarea sistemului pe unul din traseele care se termină cu un nod de blocaj. De aceea s-a elaborat procedeul care urmează pentru depistarea din timp a nodurilor de blocaj şi evitarea acestor stări în evoluția sistemului. Esența procedeului de conducere a sistemului pentru evitarea blocajelor este redată în figura următoare.

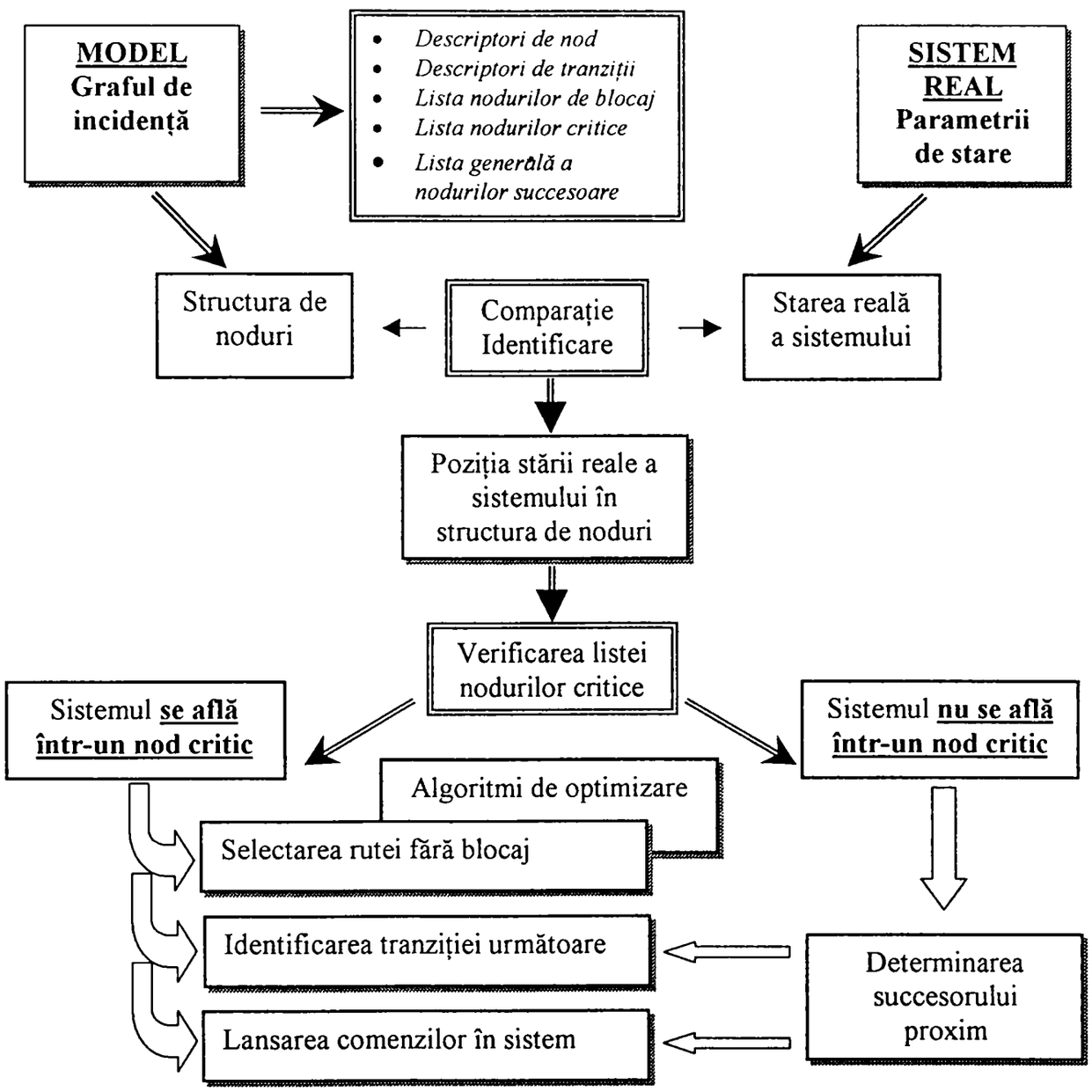


Figura 4.3-1

Pe baza schemei din Figura 4.3-1 se pot face câteva comentarii în legătură cu modalitatea concretă în care operează sistemul de conducere bazat pe evitarea blocajelor. La originea acestui sistem stă *graful de incidență*, despre care s-a vorbit în paragraful 4.2. Acest graf este prelucrat cu un sistem de funcții client care să permită scrierea unor fişiere ce vor sta

la baza dezvoltărilor ulterioare. Se scriu astfel *descriptorii de nod* ai tuturor nodurilor din sistem, care conțin stările momentane ale sistemului prin care acesta poate trece la diverse momente de timp. De asemenea se vor scrie fişierele text care conțin *descriptorii arcelor* din graful de incidență, de unde se pot extrage informații în legătură cu tranzițiile care sunt asociate fiecărui arc şi care vor duce la modificarea stării sistemului dintr-un nod în altul. In continuare, tot pe baza grafului de incidență, se va scrie *lista nodurilor de blocaj*, despre care se va vorbi mai pe larg în paragraful următor. *Nodurile critice* care se determină de asemenea pe baza grafului de incidență vor fi scrise într-o listă cu ajutorul căreia vor putea fi identificate ulterior rutele periculoase pentru evoluția sistemului, aşa-zisele "*magistrale de blocaj*". In fine, se va determina *lista tuturor succesorilor a arcelor şi a tranzițiilor* fiecărui nod din graful de incidență, necesară determinării posibilităților de evoluție ulterioară a sistemului.

Alături de datele extrase din graful de incidență se află şi *valorile determinate din sistemul real*, pe baza unei rețele de senzori şi traductori. Aceste date sunt furnizate sistemului central de conducere de către sistemele locale de control care supraveghează diversele zone de lucru din sistemul real, aşa cum se va prezenta în paragrafele următoare. Pe baza acestor informații se poate constitui o *imagine normalizată a stării sistemului*, care să poată fi comparată cu imaginile similare construite pe baza fişierelor cu descriptori de noduri şi să permită astfel identificarea uneia din stările din graful de incidență cu stare momentană reală a sistemului. Rezultă astfel numărul nodului în care se găseşte sistemul, conform numerotației din graful de incidență.

După identificare poziției sistemului în rețeaua de noduri a grafului de incidență se poate trece la *verificarea listei nodurilor critice*, pentru a vedea dacă sistemul este sau nu într-un astfel de nod. Dacă se constată că sistemul nu este într-un nod critic, se va determina succesorul proxim, adică acel nod din lista de succesori care are eticheta de timp cu valoare minimă. Această determinare permite imediat aflarea tranziției următoare şi activarea prin intermediul lanțului informațional a controlerului local responsabil cu declanşarea acesteia. Se ajunge astfel la *execuția unor comenzi în sistemul real*, pornind de la datele extrase în urma simulării. Dacă se constată că sistemul este într-un nod critic, se pune problema alegerii uneia din rutele posibile astfel încât să nu se ajungă pe o "magistrală de blocaj" de unde sistemul nu mai poate evita apariția blocajului după un anumit număr de paşi.

In faza alegerii rutei se poate interveni cu diverşi algoritmi de optimizare, aşa cum se va prezenta în paragrafele următoare. Programul de conducere a sistemului va oferi operatorului de sistem posibilitatea de a alege una din rutele posibile, eliminând automat rutele de blocaj. Figura ce urmează (Figura 4.3-2) reprezintă modul de organizare a sistemului

de conducere pentru preluarea datelor din sistemul real şi compararea lor cu datele obţinute prin simulare.

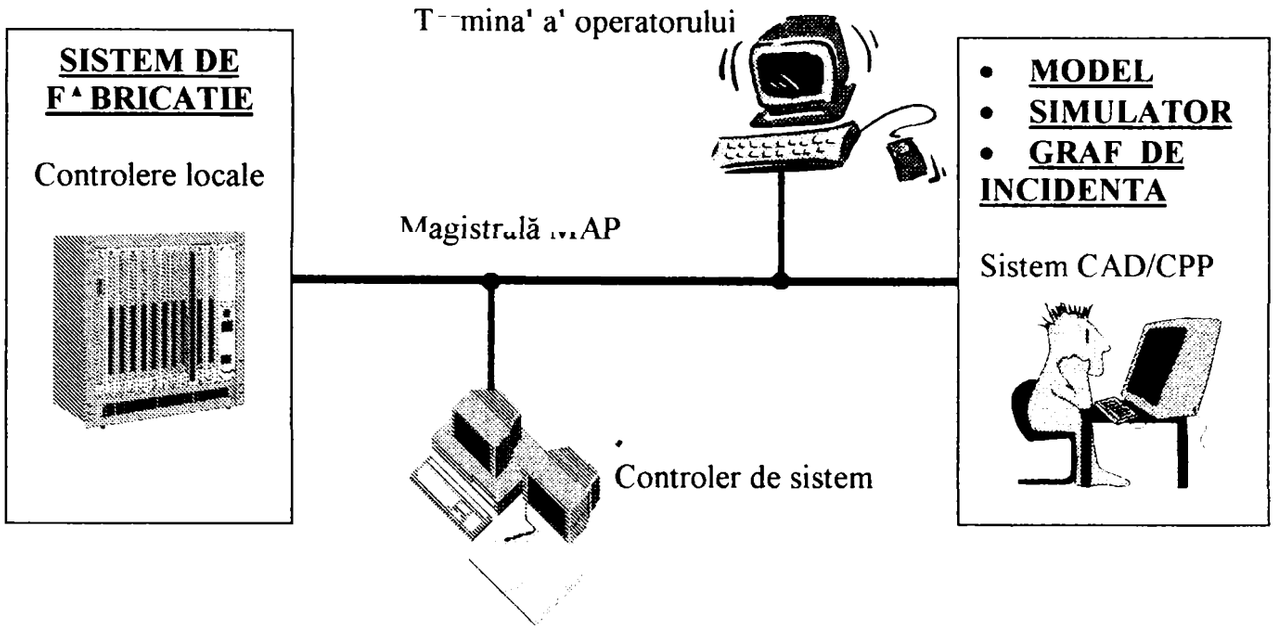


Figura 4.3-2

Simularea pe baza unui model elaborat se desfăşoară la nivelul sistemului CAD/CPP pe calculatoare care rulează de obicei un sistem de operare UNIX. Controlerul de sistem este echipamentul la nivelul căruia se va face compararea rezultatelor obţinute din simulare cu cele determinate pe baza sistemului real. Rezultatele acestor comparaţii sunt transferate către un terminal, la nivelul căruia se vor lua deciziile în legătură cu rutele de urmat. In fine, pe baza acestor decizii se va putea transmite un ordin de lucru în sistemul real, către una din zonele de lucru deservite de un controler local.

4.3.b Determinarea nodurilor de blocaj şi a magistralelor de blocaj. Scrierea descriptorilor de noduri şi arce

In cadrul modelului folosit pentru analiza sistemului a fost concepută o pagină specială destinată scrierii funcţiilor client pentru determinarea unor noduri cu proprietăţi speciale în sistem. Acestea sunt *nodurile de blocaj*, care nu mai au succesori, aflându-se la capătul unor ramuri de evoluţie a sistemului. Au fost utilizate două categorii de astfel de noduri de blocaj:

- *noduri de blocaj real*, datorat unor condiţii speciale în care se află sistemul şi care nu mai permit funcţionarea lui ulterioară;

- *noduri de blocaj fictiv*, datorat epuizării lotului de piese.

In analiza comportamentului sistemului sunt importante doar nodurile de blocaj real, deoarece blocajele care apar prin epuizarea lotului de piese sunt fireşti. In consecinţă a fost elaborat un algoritm (vezi Figura 4.3-3) şi un set de funcţii care să permită determinarea acestora şi scrierea listei lor într-un fişier text, care să poată în continuare să fie folosit pentru luarea unor decizii în legătură cu evoluţia ulterioară a sistemului.

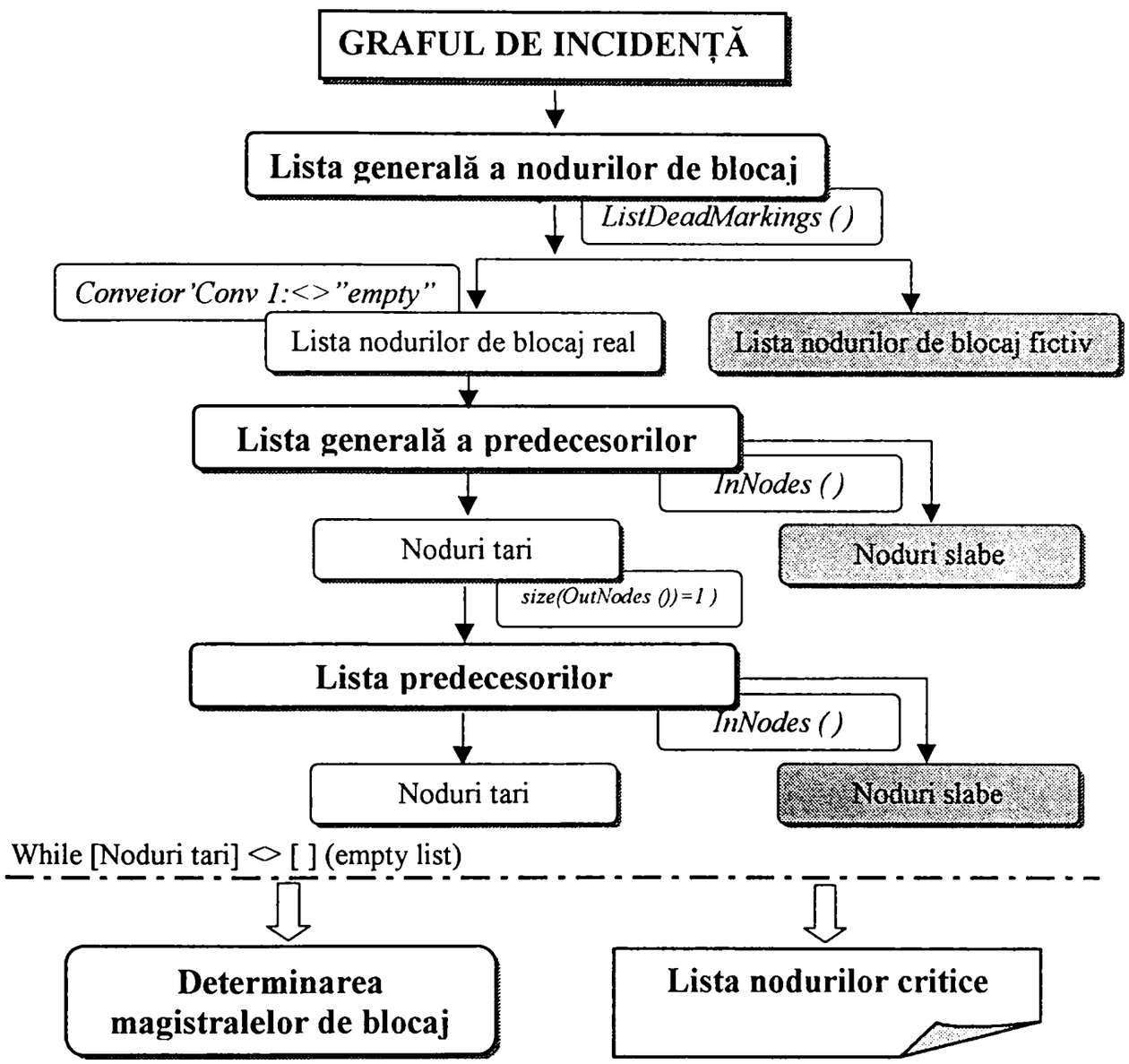


Figura 4.3-3

Algoritmul de evitare a blocajelor se bazează pe ideea de a depista succesiunile de noduri care duc în mod inevitabil la blocaj, fără nici o şansă pentru sistem de a alege o altă cale de evoluţie. Aceste succesiuni de noduri au fost denumite "*magistrale de blocaj*". Evident că orice magistrală de blocaj se va termina într-unul din nodurile aflate în lista nodurilor de blocaj. Toate nodurile aflate pe o magistrală de blocaj au proprietatea comună că

prezintă un singur succesor. Aceste noduri au fost denumite "*noduri tari*", spre deosebire de celelalte noduri care pot apare ca predecesori ai nodurilor de blocaj şi care au mai mulţi succesori, fiind denumite "*noduri slabe*". Funcţiile complete pentru determinarea nodurilor slabe şi tari precum şi a magistralelor de blocaj sunt prezentate în Anexa 4-1 şi Anexa 4-2 la prezentul capitol. De aemenea au fost prezentate funcţiile pentru determinarea listei de succesori ai nodurilor, în Anexa 4-3, precum şi programul de determinare a listei de succesori, arce şi tranziţii în Anexa 4-4.

Lista nodurilor critice care apar în figura de mai sus cuprinde toate nodurile care se află la originea unor magistrale de blocaj, sau, altfel spus, toate nodurile care se situează pe poziţia de predecesor a primului nod dintr-o magistrală de blocaj (de exemplu nodurile 240, 298 şi 359 din Figura 4.2-2, care sunt predécesorii nodurilor 316, 342 şi respectiv 377 din cadrul celor trei magistrale de blocaj prezentate). Aceste noduri critice se bucură de proprietatea că sistemul odată ajuns în această poziţie poate urma şi alte căi de dezvoltare, existând cel puţin o a doua variantă diferită de magistrala de blocaj. Nodurile critice vor fi cele care, într-o primă formă de optimizare, vor fi luate în considerare în cadrul algoritmului de conducere a sistemului de fabricaţie. **Metoda originală de conducere a sistemului real pe baza simulării constituie contribuţia autorului la optimizarea fluxului de informaţii**.

4.3.c Metodă şi algoritm de selecţie a rutelor de evitare a blocajelor

Pentru conducerea sistemului de fabricaţie pe baza rezultatelor simulării a fost conceput un program original, scris în limbajul C şi care a fost compilat pentru sistemul de operare Linux. Acest program implementează algoritmul prezentat în Figura 4.3-1.

Principala problemă de rezolvat pentru programul de conducere a fost de a găsi un format comun de prezentare atât a datelor din descriptorii de noduri cât şi a celor din sistemul real, astfel încât acestea să poată fi comparate ulterior. In această fază au fost avute în vedere două variante:

- folosirea formelor matriciale de tipul celor definite în cadrul capitolului întâi;
- definirea unei structuri în C.

Deoarece tipurile de date cu care trebuie operat sunt eterogene (date numerice şi şiruri de caractere), s-a optat pentru a doua variantă, care prezintă şi unele avantaje din punct de vedere al scrierii programului. Analizând structura unui descriptor de nod, s-au identificat un număr de parametrii care vor defini în mod univoc starea sistemului şi care vor fi incluşi în structura C. Este evident faptul că nu toţi parametrii dintr-un descriptor de nod vor face parte

din această structură. Corespondența dintre denumirile parametrilor din descriptorul de nod şi variabilele structurii, precum şi tipul acestor variabile sunt prezentate în Tabelul 4.3-1.

Tabelul 4.3-1

Numele variabilei	Poziția din model	Pagina din model	Tipul de variabilă	Observații
Dep1	Depozit1	depozit1	Int	Variabilă locală
Dep2	Dep2	depozit2	Int	Variabilă locală
Flansa1	Conv 1	conveior	Int	Variabilă locală
Flansa2	Conv 1	conveior	Int	Variabilă locală
InCloos	InCloos	sudare	Int	Variabilă locală
OutCloos	OutCloos	sudare	Int	Variabilă locală
InMF	InMF	frezare	Int	Variabilă locală
OutMF	outMF	frezare	Int	Variabilă locală
InMas	InMas	frezare	Int	Variabilă locală
OutMas	OutMas	măsurare	Int	Variabilă locală
InAsmb	InAsmb	măsurare	Int	Variabilă locală
Robot_1	Robot1	fusion place	Bool	Var. globală
Robot_2	Robot2	fusion place	Bool	Var. globală
Robot_3	Robot3	fusion place	Bool	Var. globală
AGV	AGV (AGV1)	fusion place	String	Var. globală
CtrlPs	CtrlPs	fusion place	String	Var. globală
Timp	Par. de sistem		Int	Var. globală

Codul sursă al programului "getdata" scris în C, care realizează structurarea datelor, este prezentat în Anexa 4-6 iar fişierul antet "common.h" în care a fost definită structura de date este prezentat în Anexa 4-8. Pe baza structurii obținute în acest fel este identificată starea sistemului şi se declanşează un dialog cu operatorul sau o subrutină de selecție a nodului următor (programul "control" a cărui sursă apare în Anexa 4-7) pe baza unor criterii de optimizare. Odată ce nodul următor a fost identificat, se va trece la determinarea tranziției care va declanşa modificarea de stare. In Anexa 4-4 este prezentat un extras din fişierul care conține corespondențele dintre numerele de succesori, arce şi tranziții. Deoarece aceste tranziții sunt strict legate de un anumit controler şi de o anumită pagină din model, cunoaşterea tranziției următoare va determina un număr de acțiuni asociate unui anumit post

de lucru condus de un anumit controler. In acest fel sistemul poate prelua ordine de execuţie la nivelul unui anumit controler pe baza succesiunilor de noduri şi de tranziţii din model. Identificarea tranziţiei următoare se rezolvă tot în programul C "control". Ordinograma pe baza căruia a fost construit programul "control" apare în Figura 4.3-4 şi în Figura 4.3-5.

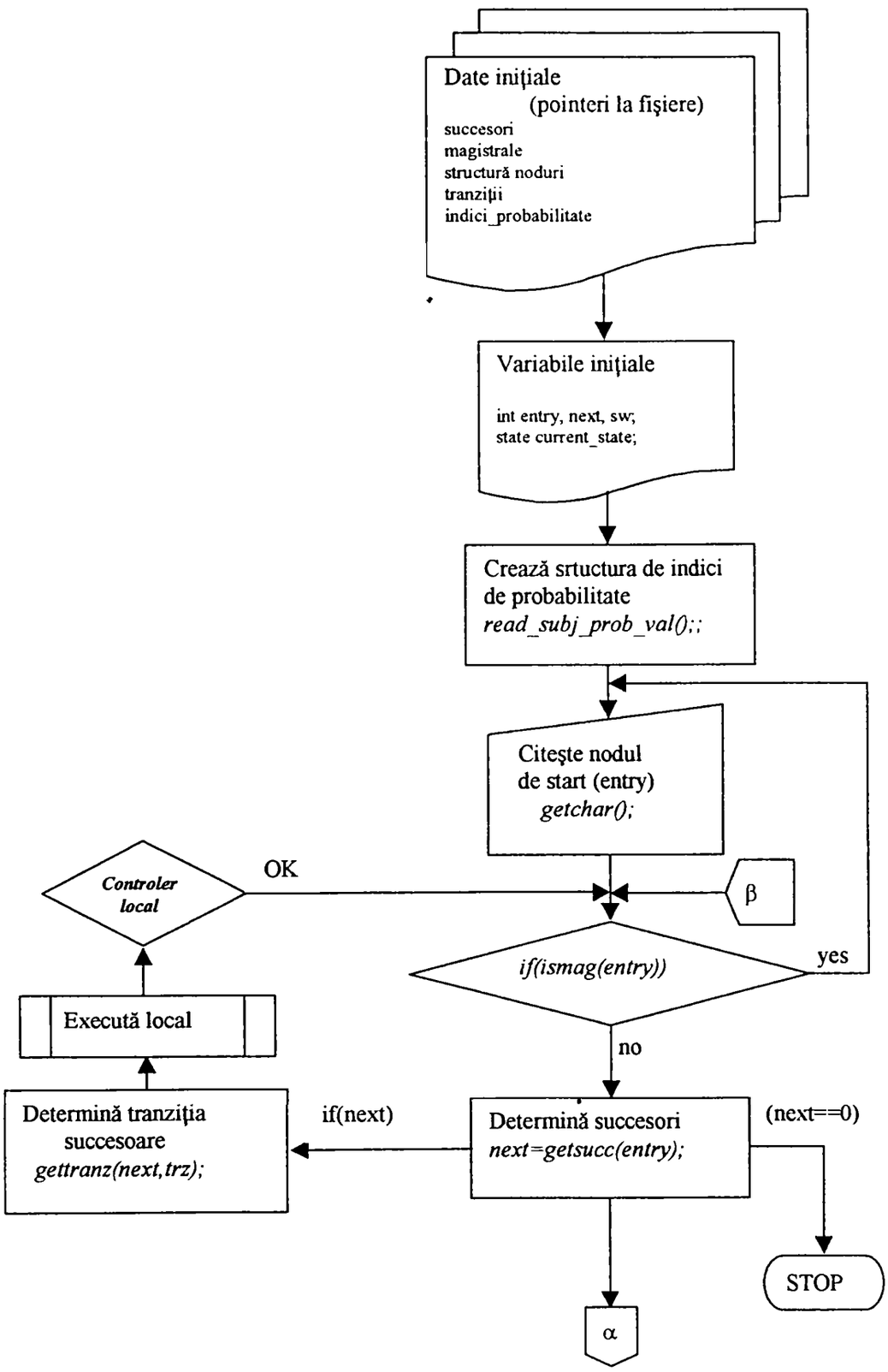


Figura 4.3-4

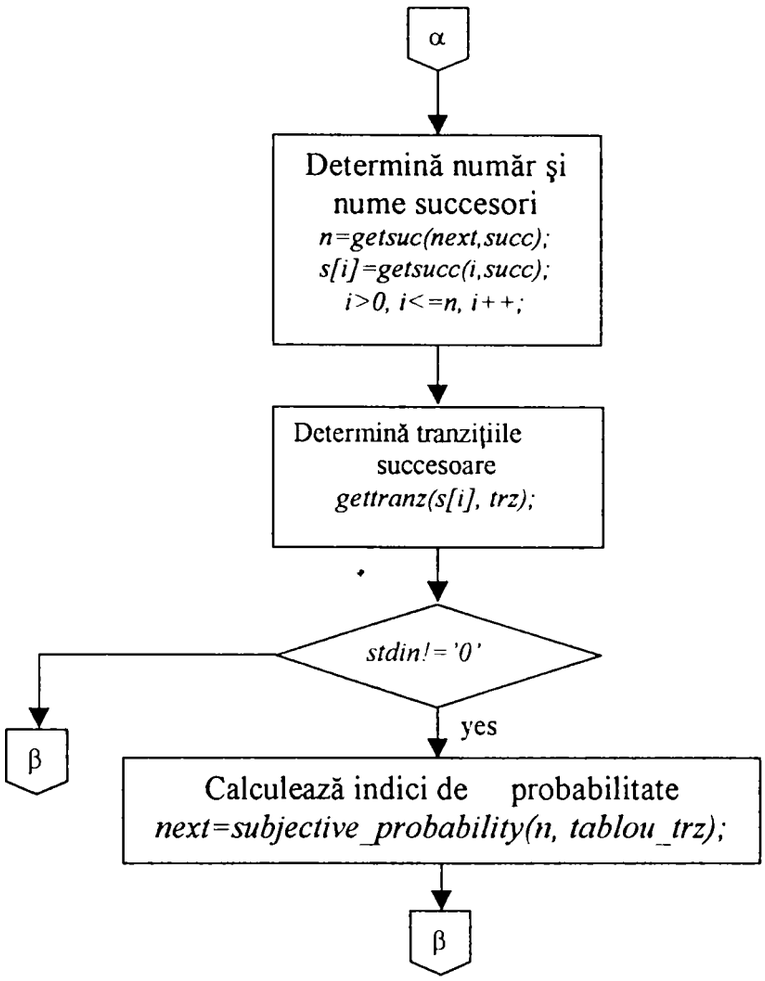


Figura 4.3-5

Pe baza numelor de tranziţii implicate într-o anumită stare de concurenţă a sistemului (mai multe tranziţii cu condiţii simultane de declanşare) se pot desfăşura diverse *rutine de optimizare a rutelor*, aşa cum se va ilustra în capitolul următor.

In componenţa sistemului de fabricaţie pentru care s-a făcut modelarea şi simularea au fost identificate un număr de *4 celule de fabricaţie independente*. Pentru fiecare dintre acestea s-au precizat un număr de variabile ale programului de conducere. Fiecare celulă este sub conducerea unui controler local, care poate fi un calculator PC, aflat în legătură cu contolerul central printr-o magistrală MAP sau Ethernet. Controlerele locale au rolul de a transmite către controlerul central unele date despre starea sistemului, care vor fi încărcate în conţinutul unor variabile ale programului de conducere central. Pe de altă parte, controlerele locale vor declanşa sarcini de execuţie locale în funcţie de deciziile luate la nivelul controlerului central. Gruparea variabilelor din structură pe cele patru celule este prezentată în tabelul următor. Pe baza valorilor variabilelor furnizate la sfârşitul fazei de lucru de către controlerele locale se poate identifica starea momentană a sistemului.

Tabelul 4.3-2

Nr. crt.	Denumirea celulei	Variabilele asociate
1.	Magazii şi depozite	Dep1, Dep2, Flansa1, Flansa2
2.	Frezare	InMF, OutMF, InMas
3.	Măsurare	OutMas, InAsmb
4.	Sudare	InCloos, OutCloos

4.3.d Propunere de arhitectură de conducere a sistemului de fabricaţie

Concepţia sistemului de conducere optimizată a unui sistem de fabricaţie a fost realizată pentru "lay-out"-ul din Figura 3.2-5 care reprezintă sistemul flexibil instalat la sediul Catedrei OMM a Facultăţii de Mecanică. Arhitectura de conducere a acestui sistem a fost concepută pornind de la cele patru celule de fabricaţie despre care se vorbeşte în § 4.3.c Aşa după cum se poate observa în Figura 4.3-6, fiecărei celule i s-a asociat un controler local (un calculator PC sau un automat programabil) care se află integrat într-o reţea locală bazată pe un protocol MAP sau TCPIP. Cele 4 controlere locale sunt în legătură cu un controler central, care rulează un sistem de operare UNIX (LinuxPC) la nivelul căruia se iau deciziile în legătură cu succesiunea nodurilor pe care le va parcurge sistemul, aşa cum apar ele în graful de incidenţă.

Tabelul 4.3-3

Sistem de conducere	Echipamente conduse
Controler1	Depozit1, Depozit2, Conveior, Robot1,AGV
Controler2	Maşina de frezat, Robot2
Controler3	Post de măsurare, Robot3
Controler4	Robot de sudare, Dispozitive de sudare

Primul echipament de conducere locală (Controler 1) este cel care se ocupă de zona depozitelor 1 şi 2 şi de conveior. Acesta monitorizează un număr de parametrii pe baza cărora pot fi setate variabilele marcate în dreptul acestui controler pe Figura 4.3-6. Aceste variabile, având tipul descris anterior în Tabelul 4.3-1, vor fi transmise controlerului central, prin intermediul protocolului de comunicaţie. Această acţiune de actualizare a variabilelor se desfăşoară ori de câte ori apar modificări în starea acestora. Echipamentele aflate sub controlul sistemului local de conducere sunt în conformitate cu Tabelul 4.3-3

Al doilea controler monitorizează şi conduce operaţiile de frezare. El primeşte informaţii din sistem pe baza cărora setează valorile variabilelor notate pe Figura 4.3-6.

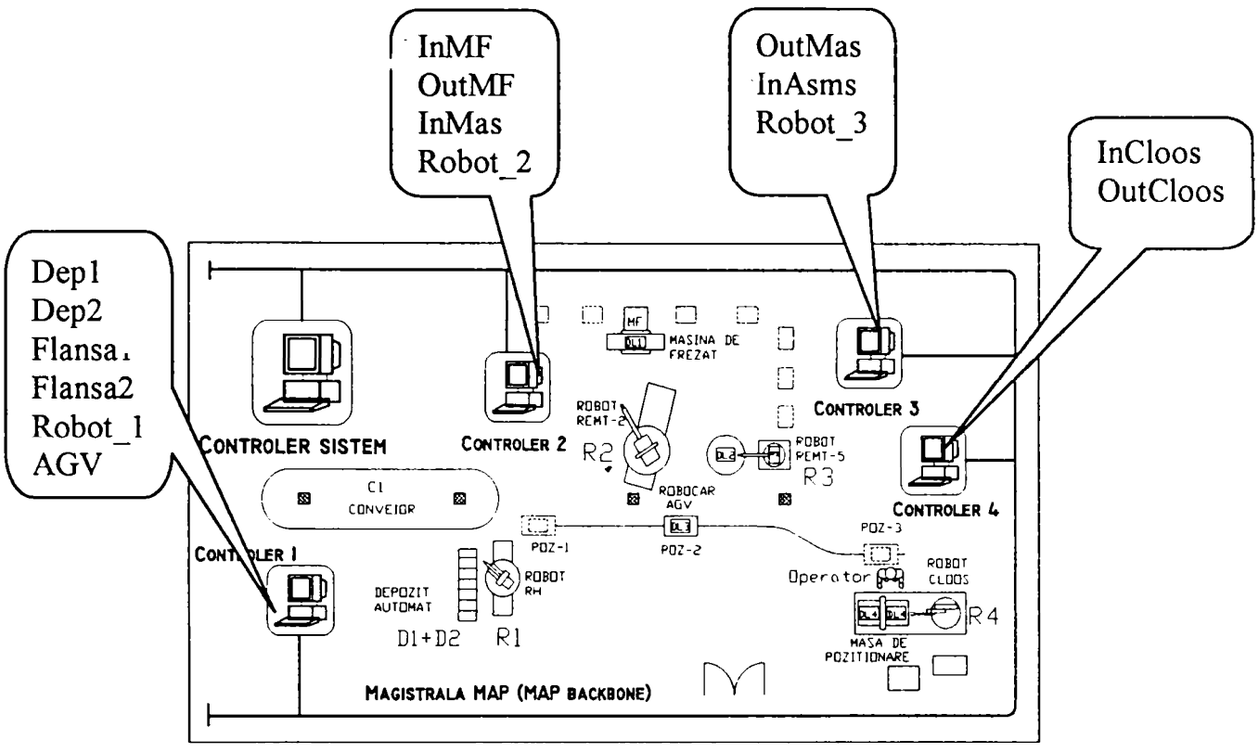


Figura 4.3-6

Tot la nivelul acestui controler sunt concentrate şi operaţiile de comandă a fazelor de încărcare/descărcare a maşinii de frezat cu ajutorul robotului 2, care execută şi operaţia de încărcare a piesei frezate pe masa de măsurare, respectiv de depunere a ei în poziţia de asamblare din AGV.

Controlerul 3 este destinat operaţiei de măsurare şi conducerii robotului 3. La nivelul acestui echipament de comandă se pot executa şi rutine specializate de prelucrare statistică a datelor.

Controlerul 4 se ocupă de operaţia de frezare şi de controlul robotului de sudare Cloos. De asemenea la acest nivel sunt prelucrate şi informaţiile legate de dispozitivele de sudare, care lucrează integrat cu robotul.

Structura sistemului de comandă se va modifica substanţial faţă de varianta prezentată în § 3.3. Pe diagrama structurală prezentată în Figura 4.3-7 se poate observa că toate echipamentele de control local sunt în contact bidirecţional cu controlerul central în care rulează programul de optimizare a rutelor de transfer, spre deosebire de cazul prezentat anterior în care o parte a sistemelor de control local (Robot R3, Robot Cloos) nu erau în nici un fel conectate cu sistemul de conducere centralizată.

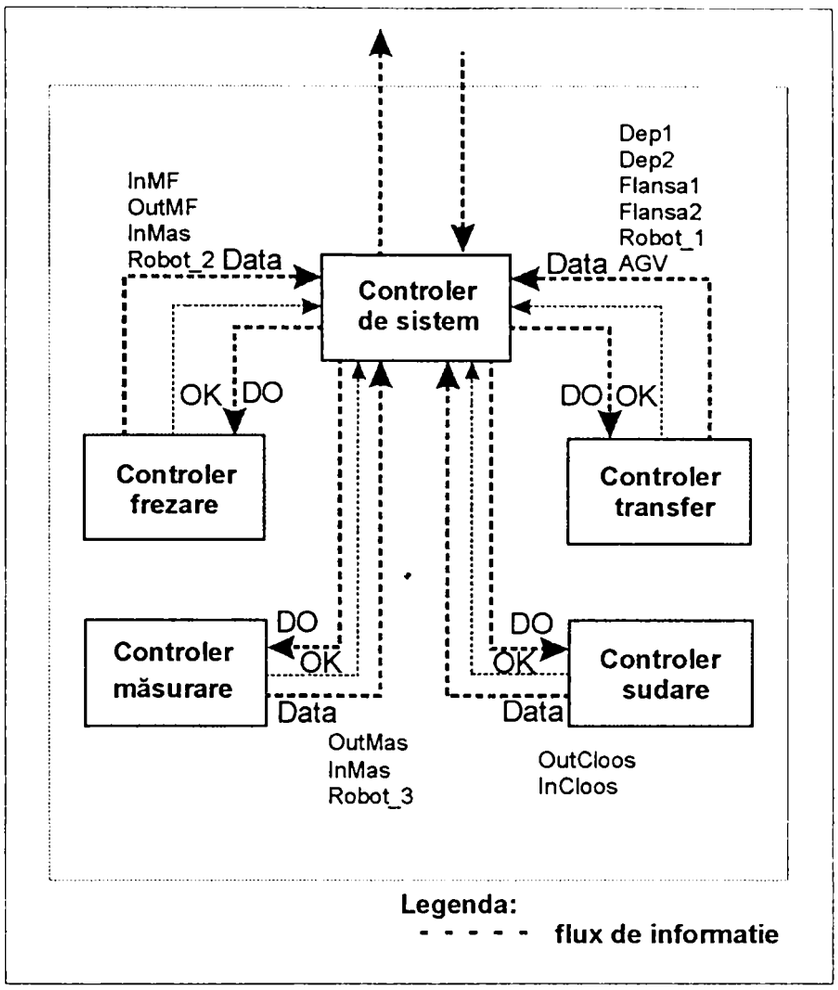


Figura 4.3-7

4.4. Descrierea încercărilor de conducere a echipamentului de frezare pe baza grafului de incidență

Pentru susținerea experimentală a modelului de conducere a sistemului de fabricație pe baza grafului de incidență a fost realizată practic legarea într-o rețea Ethernet a calculatorului central şi a unui calculator de conducere a maşinii de frezat, care a realizat o parte din funcțiile de conducere pe care le are controlerul 2 din Figura 4.3-6, în speță cele de conducere a operațiilor de încărcare/descărcare a frezei cu robotul R2, precum şi operația de descărcare din controlerul local a programului NC pentru frezarea piesei şi desfăşurarea operației propriu-zise de frezare. Acest controler de tip calculator PC a fost conceput să acopere doar sarcinile de comandă la nivelul celulei de frezare, prin controlul axelor robotului şi a descărcării programelor de frezare. Deoarece sistemele de acționare ale maşinii de frezat nu au putut fi integrate în sistemul general de comandă al axelor robotului, s-a recurs la comanda maşinii de frezat prin intermediul echipamentului de comandă NC. Ciclograma operațiilor desfăşurate la acest nivel este prezentată în figura următoare.

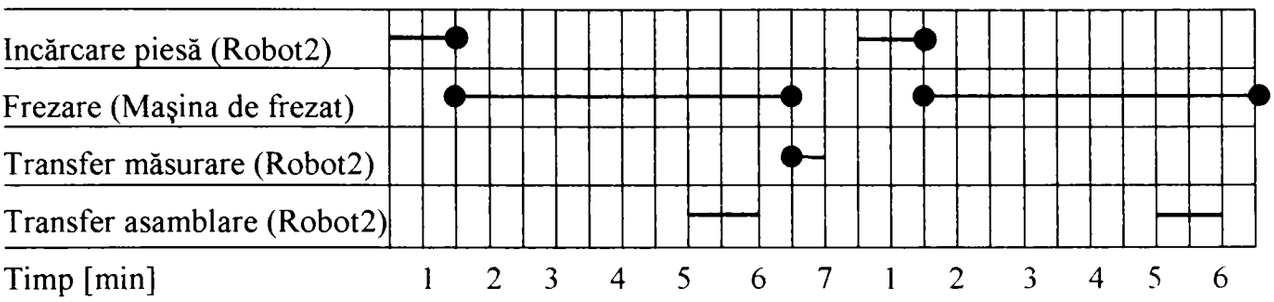


Figura 4.4-1

Calculatorul central detectează operația de frezare ca operație următoare în ciclu, pe baza grafului de incidență şi a algoritmului de decizie prezentat în paragrafele anterioare. In acest moment controlul este transferat calculatorului de conducere al celulei de frezare. Acesta declanşează mai întâi secvența de program referitoare la controlul axelor robotului de servire al maşinii, "R2". Acesta primeşte o comandă de deplasare pe una din axe prin intermediul unei interfețe speciale a calculatorului construită în acest scop, pe care se preia şi semnalul de răspuns de la traductorul TIRO cu care este echipată fiecare axă a robotului. Mişcările pe cele 4 axe ale robotului sunt conduse pe baza programului, urmărindu-se pozițiile cu ajutorul traductorilor. Inaintea finalului operației de încărcare se dă comanda de închidere pneumatică a bacurilor dispozitivului de fixare a piesei pe masa maşinii de frezat.

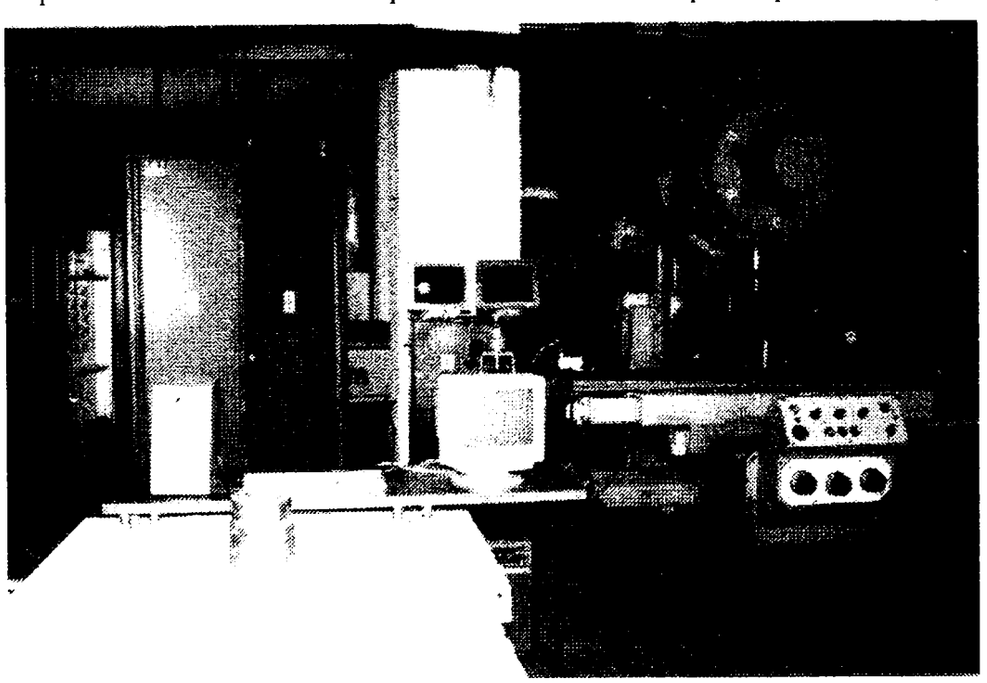


Figura 4.4-2

După terminarea operației de încărcare a piesei, din acelaşi calculator se execută un program de postprocesare a frazelor NC în comenzi pe 8 biți transferate la portul paralel al calculatorului, care este conectat la sistemul de comandă al frezei. Programul se află stocat pe

hard disk-ul calculatorului de comanda în format text. Acest program a fost obţinut în prealabil cu un soft specializat pentru descrierea geometrică şi generarea traiectoriilor pe maşini cu comandă numerică denumit MANA. După transferarea completă a programului NC către maşina de frezat, se declanşează mişcările de evacuare a piesei cu ajutorul robotului R2 către centrul de măsurare, unde este preluată de robotul R3 aflat sub conducerea altui controler. Programul care rulează în controlerul local al celulei de frezare a fost realizat de colectivul care a proiectat şi realizat sistemul de fabricaţie şi a fost folosit de autor doar pentru validarea ipotezelor teoretice, fără să i se aducă nici un fel de modificări. Elementul de interes a fost constituit doar de declanşarea acestui program de control local de la nivelul unei instanţe superioare ierarhic pe linie de comandă a sistemului şi de returnarea unei valori de control din acest program către controlerul central atunci când execuţia lui se termină.

In figura Figura 4.4-2 este redată fotografia celulei de prelucrare prin frezare folosită la efectuarea încercărilor.

4.5. Anexe

Anexa 4-1 Funcțiile de determinare a nodurilor slabe și tari în limbaj CpnMl

```
(*AFISAREA SI INREGISTRAREA TUTUROR NODURILOR DE BLOCAJ*)
fun ListDeadMarkings () : Node list
= PredAllNodes Terminal;
ListDeadMarkings();
let
val outfile=open_out
"/home/mircea/modele/modele_teza/noduri/noduri_blocaj";
val lista=(ListDeadMarkings());
fun CPN'Print [] CPN'n = "" (* None will be printed *)
                  | CPN'Print (CPN'el1::CPN'rest) CPN'n =
                    mkst_Node(CPN'el1)^" "^(CPN'Print CPN'rest (CPN'n+1));
in
(output (outfile, (CPN'Print (lista) 1)^ "\t");
close_out (outfile))
end;
(*INREGISTRAREA NODURILOR DE BLOCAJ CARE NU SE DATORESC LIPSEI  DE PIESE
(Blocaje reale)*)
fun SelectieNoduri_blocaj_real ()
=
SearchNodes (
EntireGraph,
fn n => Terminal n  andalso   (st_Mark.conveior'Conv 1 n<> "conveior'Conv
1: tempty\n" ),
NoLimit,
fn n=> n,
[],
op ::);
let
val outfile=open_out
"/home/mircea/modele/modele_teza/noduri/noduri_blocaj_real";
val lista=(SelectieNoduri_blocaj_real());
fun CPN'Print [] CPN'n = "" (* None will be printed *)
                  | CPN'Print (CPN'el1::CPN'rest) CPN'n =
                    mkst_Node(CPN'el1)^" "^(CPN'Print CPN'rest (CPN'n+1));
in
(output (outfile, (CPN'Print (lista) 1)^ "\t");
close_out (outfile))
end;
(*INREGISTRAREA NODURILOR DE BLOCAJ CARE SE DATORESC LIPSEI  DE PIESE
(Opriri la capat de lot)*)
fun SelectieNoduri_lot ()
=
SearchNodes (
EntireGraph,
fn n => Terminal n  andalso   (st_Mark.conveior'Conv 1 n= "conveior'Conv 1:
tempty\n" ),
NoLimit,
fn n=> n,
[],
op ::);
let
val outfile=open_out "/home/mircea/modele/modele_teza/noduri/noduri_lot";
val lista=(SelectieNoduri_lot());
fun CPN'Print [] CPN'n = "" (* None will be printed *)
                  | CPN'Print (CPN'el1::CPN'rest) CPN'n =
```

```
                    mkst_Node(CPN'el1)^" "^(CPN'Print CPN'rest (CPN'n+1));
in
(output (outfile, (CPN'Print (lista) 1)^ "\t");
close_out (outfile))
end;
(*INREGISTRAREA NODURILOR PREDECESOARE NODURILOR DE BLOCAJ REAL*)
fun SelectieNoduri_blocaj_real1 ()
=
SearchNodes (
SelectieNoduri_blocaj_real(),
fn n => true ,
NoLimit,
fn n => InNodes(n),
[],
op ^^);
let
val outfile=open_out "/home/mircea/modele/modele_teza/noduri/noduri_1";
val lista=(SelectieNoduri_blocaj_real1());
fun CPN'Print [] CPN'n = "" (* None will be printed *)
                  | CPN'Print (CPN'el1::CPN'rest) CPN'n =
                    mkst_Node(CPN'el1)^" "^(CPN'Print CPN'rest (CPN'n+1));
in
(output (outfile, (CPN'Print (lista) 1)^ "\t");
close_out (outfile))
end;
(*INREGISTRAREA NODURILOR TARI*)
fun SelectieNoduri_1tari ()
=
SearchNodes (
SelectieNoduri_blocaj_real1(),
fn n =>(length(InArcs(n)) <= 1 andalso length(OutArcs(n)) =1),
NoLimit,
fn n=> n,
[],
op ::);
let
val outfile=open_out "/home/mircea/modele/modele_teza/noduri/noduri_1tari";
val lista=(SelectieNoduri_1tari());
fun CPN'Print [] CPN'n = "" (* None will be printed *)
                  | CPN'Print (CPN'el1::CPN'rest) CPN'n =
                    mkst_Node(CPN'el1)^" "^(CPN'Print CPN'rest (CPN'n+1));
in
(output (outfile, (CPN'Print (lista) 1)^ "\t");
close_out (outfile))
end;
(*INREGISTRAREA NODURILOR SLABE*)
fun SelectieNoduri_1slabe ()
=
SearchNodes (
SelectieNoduri_blocaj_real1(),
fn n => (length(InArcs(n)) >1 orelse length(OutArcs(n)) >1),
NoLimit,
id,
[],
op ::);
let
val outfile=open_out
"/home/mircea/modele/modele_teza/noduri/noduri_1slabe";
val lista=(SelectieNoduri_1slabe());
fun CPN'Print [] CPN'n = "" (* None will be printed *)
                  | CPN'Print (CPN'el1::CPN'rest) CPN'n =
```

```
                    mkst_Node(CPN'el1)^" "^(CPN'Print CPN'rest (CPN'n+1));
in
(output (outfile, (CPN'Print (lista) 1)^ "\t");
close_out (outfile))
end;
```

Anexa 4-2 Funcțiile de determinare a magistralelor de blocaj în limbaj CpnML

```
(*DETERMINAREA MAGISTRALELOR DE BLOCAJ*)
fun MagistraleBlocaj_string()
=
let
val c=316;
in
SearchNodes (
SelectieNoduri_blocaj_real(),
fn n => Reachable (c,n) ,
NoLimit,
fn n =>mkst_Node(n),
"",
op  ^)
end;
MagistraleBlocaj_string();
fun MagistraleBlocaj_nod()
=
let
val c=316;
in
SearchNodes (
SelectieNoduri_blocaj_real(),
fn n => Reachable (c,n) ,
NoLimit,
fn n =>n,
[],
op ::)
end;
MagistraleBlocaj_nod();
(*fun TransformaNod_str ()
=
mkst_ms'numere (list_to_ms(MagistraleBlocaj_nod()));
TransformaNod_str();*)
fun ScrieMagistrale()
=
let
val outfile=open_append
"/home/mircea/modele/modele_teza/noduri/magistrale";
fun CPN'Print [] CPN'n = "" (* None will be printed *)
                  | CPN'Print (CPN'el1::CPN'rest) CPN'n =
                    mkst_Node(CPN'el1)^" "^(CPN'Print CPN'rest (CPN'n+1));
fun Magistrale()
=
let
val a=MagistraleBlocaj_nod();
in
NodesInPath(316,hd(a))
end;
in
(output (outfile, (CPN'Print (Magistrale()) 1)^ "\n");
close_out (outfile))
end;
ScrieMagistrale();
```

Anexa 4-3 Programul pentru determinarea listei de succesori a tuturor nodurilor din graful de incidență în limbaj CpnML

```
fun GenerareSuccesori ()
=
SearchNodes (
EntireGraph,
(*PredAllNodes Terminal ,*)
fn n => true,
NoLimit,
fn n =>
(let
val lista=OutNodes(n);
val fisier="/home/mircea/modele/modele_teza/noduri/succesori";
val outfile=open_append fisier;
fun CPN'Print [] CPN'n = "" (* None will be printed *)
                  | CPN'Print (CPN'el1::CPN'rest) CPN'n =
                    mkst_Node(CPN'el1)^" "^(CPN'Print CPN'rest (CPN'n+1));
in
(output (outfile, mkst_col'numere (n)^"\t"^ (CPN'Print (lista) 1)^"\n");
close_out (outfile))
end),
1,
fn ((),int) => 1+1);
GenerareSuccesori ();
```

Anexa 4-4 Fragment din lista de succesori, arce şi tyranziţii ale tuturor nodurilor din sistem.

```
2      1      conveior'TriAGV
9      10     masurare'Masurare
62     100    frezare'TriMF
62     101    masurare'Masurare
61     102    masurare'Masurare
61     103    masurare'Masurare
61     104    masurare'Masurare
61     105    masurare'Masurare
61     106    masurare'Masurare
61     107    masurare'Masurare
61     108    masurare'Masurare
61     109    masurare'Masurare
9      11     masurare'Masurare
61     110    masurare'Masurare
63     111    frezare'Frez
64     112    conveior'TriAGV
65     113    asamblare'Asmb
66     114    frezare'Frez
67     115    sudare'TroCloos
68     116    frezare'Frez
69     117    sudare'TroCloos
70     118    conveior'TriAGV
71     119    frezare'Frez
145    272    masurare'Masurare
145    273    masurare'Masurare
145    274    masurare'Masurare
145    275    masurare'Masurare
145    276    masurare'Masurare
145    277    masurare'Masurare
145    278    masurare'Masurare
145    279    masurare'Masurare
19     28     masurare'Masurare
146    280    sudare'TriCloos
147    281    depozit2'TrDep2
148    282    sudare'TriCloos
149    283    frezare'TriMF
150    284    frezare'TriMF
150    285    masurare'Masurare
149    286    masurare'Masurare
149    287    masurare'Masurare
149    288    masurare'Masurare
149    289    masurare'Masurare
20     29     masurare'Masurare
149    290    masurare'Masurare

203    383    depozit1'TrDep1
204    384    asamblare'Asmb
205    385    asamblare'Asmb
206    386    depozit2'TrDep2
207    387    depozit2'TrDep2
208    388    depozit1'TrDep1
209    389    depozit1'TrDep1
22     39     sudare'TriCloos
5      4      conveior'TriAGV
23     40     sudare'Sudare
24     41     sudare'TroCloos
25     42     sudare'TroCloos
26     43     conveior'TriAGV
25     44     sudare'Sudare
27     45     depozit1'TrDep1
```

Anexa 4-5 Programul de determinare a listei de succesori arce şi tranziţii ale tuturor nodurilor idn sistem în limbaj CpnMl

```
fun GenerareNoduri_Arce ()=
SearchArcs (
EntireGraph,
(*PredAllNodes Terminal ,*)
fn n => true,
NoLimit,
fn n =>
(let
val lista=DestNode(n);
val lista1=st_TI(ArcToTI(n));
val fisier="/home/mircea/modele/modele_teza/noduri/succ_arce_trz";
val outfile=open_append fisier;
fun CPN'Print CPN'n =
                    mkst_Node(CPN'n);
in
(output (outfile, CPN'Print (lista)^"\t"^mkst_col'numere
(n)^"\t:"^lista1^"\n");
close_out (outfile))
end),
1,
fn ((),int) => 1+1);
GenerareNoduri_Arce ();
```

Anexa 4-6 Programul C "getdata" pentru colectarea datelor pe baza descriptorilor de noduri

```
#include <sys/types.h>
#include <sys/stat.h>

#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <dirent.h>
#include <string.h>
#include <stdlib.h>

#include "common.h"

FILE *out;
DIR  *dir;

      int cleanup(int error);
      int newrecord(FILE *in);
      int get_state_number(FILE *in);
      char *get_param_data(const char *name,FILE *in);
      int get_param_data_int(const char *name,FILE *in);
      int getmaxtime(FILE *in);

int main(int argn,char *argc[]) {
  struct dirent *file;
  char   *a;
  DIR    *try;
  FILE   *in;

  if(argn!=3) { printf(" Usage: getdata <nodes directory> <destination
file>"); cleanup(1); }
  if(!(dir=opendir(argc[1]))) { printf(" Error !!! Can't open dir %s
",argc[1]); cleanup(1); }
  if(!(out=fopen(argc[2],"w"))) { printf(" Error !!! Can't open file <%s>
for write acess",argc[2]); cleanup(1);}
  file=readdir(dir);
  while(file!=NULL) {
      a=malloc(strlen(argc[1])+strlen(file->d_name)+2);
        memcpy(a,argc[1],strlen(argc[1]));
        a[strlen(argc[1])]='/';
        memcpy(&a[strlen(argc[1])+1],file->d_name,strlen(file->d_name)+1);
        if(!(try=opendir(a))) {
            printf("\n %s",file->d_name);
                  in=fopen(a,"r");
                  newrecord(in);
        }
        else closedir(try);
      file=readdir(dir);
  }
  printf("\n");
  cleanup(0);
 return(0);
}

int cleanup(int error) {
  closedir(dir);
  fclose(out);
  if(error) { printf("\n");exit(1);}
```

```
 return(0);
}

int newrecord(FILE *in) {
  state new;
  char *dd;
  int i;

  new.nr=get_state_number(in);
  new.Robot1=get_param_data_int("depozit1'Robot1 1",in);
  new.Robot2=get_param_data_int("frezare'Robot2 1" ,in);
  new.Robot3=get_param_data_int("masurare'Robot3 1",in);

  new.Tmax=getmaxtime(in);

  dd=get_param_data("depozit1'AGV 1",in);
  strcpy(new.AGV,dd);
  free(dd);
  dd=get_param_data("depozit1'CtrlPs 1",in);
  strcpy(new.CtrlPS,dd);
  free(dd);

  new.comp1.Dep1=get_param_data_int("depozit1'Depozit1 1",in);
  new.comp1.Dep2=get_param_data_int("depozit2'Dep2 1",in);
  new.comp1.Flansa=get_param_data_int("conveior'Conv 1",in);

  dd=get_param_data("conveior'Conv 1",in);
  i=0;
  while(dd[i]!=0x60) i++;
  new.comp1.FlansaN=dd[i+9];
  free(dd);

  new.comp2.InMF=get_param_data_int("frezare'InMF 1",in);
  new.comp2.OutMF=get_param_data_int("frezare'outMF 1",in);
  new.comp2.InMas=get_param_data_int("frezare'InMas 1",in);

  new.comp3.OutMas=get_param_data_int("masurare'OutMas 1",in);
  new.comp3.Asms=get_param_data_int("masurare'InAsmb 1",in);

  new.comp4.InCloos=get_param_data_int("sudare'InCloos 1",in);
  new.comp4.OutCloos=get_param_data_int("sudare'OutCloos 1",in);

  fwrite(&new,sizeof(new),1,out);
  return(0);
}

int get_state_number(FILE *in) {
  char line[256];
  rewind(in);
  fgets(line,255,in);
  return(atoi(line));
}

char *get_param_data(const char *name,FILE *in) {
  char *line;
  char *data=NULL;
  int i;
  line=malloc(256);
  rewind(in);
  do {
   fgets(line,255,in);
```

```
   i=0;
   while(line[i]!=':' && i!=strlen(line)) i++;
   line[i]=0;
  } while(!feof(in) && strcmp(name,line));
  if(strcmp(name,line)==0) {
    data=malloc(strlen(&line[i+1])+2);
      strcpy(data,&line[i+1]);
      free(line);
  }
  return(data);
}

int get_param_data_int(const char *name,FILE *in) {
  char *dd;
  int i;
  dd=get_param_data(name,in);
  i=0;
  while(dd[i]!=0x27 && i!=strlen(dd)-1) i++; dd[i]=0x00;
  if(i==strlen(dd)-1 && dd[i]!=0x27) return(0);
  return(atoi(dd));
}

int getmaxtime(FILE *in)
{
  int i,j;
  int s;
  int tmax=0;
  int ttest;
  char line[256];
  rewind(in);
  do {
   fgets(line,255,in);
   i=0;
   while(line[i]!='[' && i!=strlen(line)) i++;
   if(i!=strlen(line)) {
      i=i+1;
        s=strlen(line);
      for(j=i;j<s;j++) {
         if(line[j]==',' || line[j]==']') {
                line[j]=0x00;
                  ttest=atoi(&line[i]);
                  if(tmax<ttest) tmax=ttest;
                  i=j+1;
            }
      }
  }
  } while(!feof(in));
  printf(" %i ",tmax);
  return(tmax);
}
```

Anexa 4-7 Programul C "control" folosit la luarea deciziilor pentru evitarea blocajelor.

```
#include <sys/types.h>
#include <sys/stat.h>

#include <stdio.h>
#include <fcntl.h>
#include <dirent.h>
#include <string.h>
#include <stdlib.h>

#include "common.h"

FILE *mag;
FILE *suc;
FILE *db;

      int cleanup(int error);
      int getsucc(int entry);
      int ismag(int nod);
      int getsuc(int nod,int *s);
      int sujective_probabily(int n,int *nodes);
      int getdata(state *ndata,int nod);

int main(int argn,char *argc[]) {

  int entry;
  int next;
  int sw;
  state current_state;

  if(argn!=4) { printf(" Usage: getdata <succ_file> <mag_file> <db_file>
"); cleanup(1); }
  if(!(suc =fopen(argc[1],"r"))) { printf(" Error !!! Can't open file <%s>
for read acess",argc[2]); cleanup(1);}
  if(!(mag =fopen(argc[2],"r"))) { printf(" Error !!! Can't open file <%s>
for read acess",argc[3]); cleanup(1);}
  if(!(db  =fopen(argc[3],"r"))) { printf(" Error !!! Can't open file <%s>
for read acess",argc[4]); cleanup(1);}
  printf("\n Enter the entry point : ");scanf("%i",&entry);
  fflush(stdout);
  getchar();
  sw=0;
  while(sw==0) {
        printf("\n Current node : % i",entry);
        if(getdata(&current_state,entry)) printf("\n         Extended node
data available in 'current_state' variable");

        next=getsucc(entry);
        if(next==0) {
            printf("\n\n Failure !!! Node %i is a 'mag' node or has no
suitable 'succ'",entry);
            cleanup(1);
        }
        printf("         Next node    : %i" ,next);
        printf("\n         Press the Enter key for next node ... ");
        getchar();
        entry=next;
        next=0;
  }
```

```
  printf("\n");
  cleanup(0);
 return(0);
}

int cleanup(int error) {
  fclose(suc);
  fclose(mag);
  fclose(db);
  if(error) { printf("\n");exit(1);}
 return(0);
}

int getsucc(int entry) {
 char line[256];
 int s[200];
 int n;
 int i;
 int choice;

 if(ismag(entry)) return(0);
 n=getsuc(entry,s);
 if(n==1) { printf("\n"); return(s[0]); }
 if(n>1) {
   printf("\n         Detected More than one jump possible ... ");
   printf("\n         Please select one of these possibilities : ");
   printf("\n         0) subjective probability ... ");
   printf("\n        ");

   for(i=0;i<n;i++) printf(" %i) %i ",i+1,s[i]);
   printf("\n         Your choice : ");scanf("%i",&choice);
   getchar();
   if(choice>0 && choice<=n) return(s[choice-1]);
   else return(sujective_probabily(n,s));
 }
 return(0);
}

int ismag(int nod) {
  char line[256];
  int  i;
  int comp;
  rewind(mag);
  while(!feof(mag)) {
    fgets(line,255,mag);
      i=0;
      while(line[i]>=0x30 && line[i]<=0x39 && i<strlen(line)) i++;
      line[i]=0x00;
      comp=atoi(line);
      if(comp==nod) return(1);
  }
 return(0);
}

int getsuc(int nod,int *s) {
  char line[256];
  int comp,sl;
  int sw=0;
  int n=0;
  int i,j;
```

```
  rewind(suc);
  while(!feof(suc) && sw==0) {
      fgets(line,255,suc);
      sl=strlen(line);
      i=0;j=0;
      while(line[i]>=0x30 && line[i]<=0x39 && i<sl) i++;
      line[i]=0x00;
      comp=atoi(line);
      i=i+1;
      if(comp==nod) {
         while(i<sl && j<sl) {
            while((line[i]<0x30 || line[i]>0x39 ) && i<sl) i++;
            j=i;
            while(line[j]>=0x30 && line[i]<=0x39 && j<sl) j++;
            line[j]=0x00;
            s[n]=atoi(&line[i]);
            if(!ismag(s[n])) n++;
            i=j+1;
         }
      }
  }
  return(n);
}

int getdata(state *ndata,int nod) {
  state test;
  rewind(db);
  test.nr=0;
  while(!feof(db) && test.nr!=nod) {
      fread(&test,sizeof(state),1,db);
  }
  if(nod==test.nr) {
    memcpy(ndata,&test,sizeof(state));
    return(1);
  }
 return(0);
}

int sujective_probabily(int n,int *nodes)
{
 int i;

 printf("\n           Subjective routine: Total number of elements :
%i",n);
 printf("\n           Subjective routine: ");
 for(i=0;i<n;i++) printf(" %i ",nodes[i]);
 return(0);
}
```

Anexa 4-8 Fişierul antet "common.h" pentru definirea structurii de date C pe baza descriptorilor de noduri

```
#define bool unsigned char

typedef struct {
   int Dep1;
   int Dep2;
   char FlansaN;
   int Flansa;
} control_1;

typedef struct {
   int InMF;
   int OutMF;
   int InMas;
} control_2;

typedef struct {
   int OutMas;
   int Asms;
} control_3;

typedef struct {
  int InCloos;
  int OutCloos;
} control_4;

typedef struct {
   int  nr;
   bool Robot1;
   bool Robot2;
   bool Robot3;
   char AGV[50];
   char CtrlPS[50];
   int  Tmax;
   control_1 comp1;
   control_2 comp2;
   control_3 comp3;
   control_4 comp4;
} state;
```

4.6. Bibliografie

1. ***,1993, Design/CPN Reference Manual for X-Windows.Version 2.0, Meta Sofware Corporation, Cambridge, MA.
2. Bulac, C., 1995, Inițiere în Turbo C++ şi Borland C, seria Calculatoare personale, Editura Teora, Bucureşti.
3. Jean-Marie Proth, Ioannis Minis, 1995, Planning and Scheduling based on Petri Nets. In "Petri Nets in Flexible and Agile Automation", Kluewer Academic Publishers, Boston , Dordrecht, London.
4. Jensen Kurt, 1994, Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 2, Analysis Methods. Springer-Verlag, Berlin.
5. Keyi Xing, Boosheng Hu, Haoxun Chen, 1995, Deadlock Avoidance Policy for Flexible Manufacturing Systems. In "Petri Nets in Flexible and Agile Automation", Kluewer Academic Publishers, Boston , Dordrecht, London.
6. Kovacs, Fr., Cojocaru, G., 1982, Manipulatoare, roboți şi aplicațiile lor industriale. Editura Facla, Timişoara.
7. Kovacs, Fr., Rădulescu, C., 1992, Roboți industriali, Vol. I şi II, Litografia Universității "Politehnica" Timişoara.
8. Luca Ferrarini, 1995, Computer Aided Design of Logic Controllers with Petri Nets. In "Petri Nets in Flexible and Agile Automation", Kluewer Academic Publishers, Boston , Dordrecht, London.
9. Nof, S., 1992, Handbook of Industrial Robotics. Krieger Publishing Company, Malabar, Florida.
10. Popovici, D., M., Popovici, I., M., Tănase, I., 1996, Tehnologia orientată p eobiecte. Aplicații. Editura Teora, Bucureşti.
11. Shammas, N., C., 1996, Curs rapid de Borland C++ 4, Editura Teora, Bucureşti.
12. Tomoshiro Murata, 1995, Application of Petri Nets to Sequence Control Programming. In "Petri Nets in Flexible and Agile Automation", Kluewer Academic Publishers, Boston , Dordrecht, London.

5. AUTOMATIZAREA ŞI OPTIMIZAREA LUĂRII DECIZIEI PRIVIND TRANSFERUL MATERIALELOR ÎN SISTEME DE FABRICAȚIE PE BAZA NOȚIUNII DE PROBABILITATE SUBIECTIVĂ

5.1. Cuprins

5.2. Considerații teoretice

Prezentarea teoremelor lui Bayes [Hietiko, E., 1996, pag. 36 şi urm.] şi folosirea lor în modelarea sistemelor cu evenimente discrete are drept scop eliminarea *factorilor de incertitudine* care apar în dezvoltarea simulării. Asemenea incertitudini manifestate în procesul de luare a deciziilor se rezolvă în lumea reală prin aportul inteligenței umane sau prin acțiunea hazardului. In cazul modelării sistemelor continui, eliminarea gradelor de incertitudine se face folosind rețele neuronale sau tehnicile "fuzzy". In cazul sistemelor cu evenimente discrete, schimbările de stare ale sistemului se produc ca urmare a unor "acumulări" de condiții favorabile şi totodată necesare, acumulări ce pot fi constatate în nodurile rețelei şi care pot favoriza declanşarea simltană a mai multor tranziții. Pentru a introduce o oarecare logică în luarea deciziei şi a controla în acest fel dezvoltarea simulării, se propune utilizarea conceptului de "*probabilitate subiectivă*" precum şi a altor concepte înrudite, utilizarea acestora făcându-se pe baza teoremelor lui Bayes.

Teoria probabilităților subiective nu operează doar cu *evenimente* care au o anumită probabilitate de a se produce, ci şi cu *propoziții sau ipoteze*, a căror credibilitate creşte pe măsură ce creşte încrederea într-o evidență. In termenii teoriei probabilității subiective, dacă notăm cu "**A**" o anumită propoziție (ipoteză), atunci "**P(A)**" va fi gradul de încredere individuală în veridicitatea ipotezei respective.

5.2.a Relațiile de calcul a indicilor subiectivi de probabilitate

Se presupune că pentru un eveniment oarecare există "n" propoziții legate de desfăşurarea lui, propoziții mai mult sau mai puțin adevărate. Astfel, conform celor de mai sus, "$\mathbf{P(z_i)}$" va însemna probabilitatea ca propoziția "$\mathbf{z_i}$" să fie adevărată. Această valoare este definită ca *probabilitatea primară* a propoziției "$\mathbf{z_i}$". Orice evidență "**E**" asociată propoziției "$\mathbf{z_i}$" va modifica nivelul de încredere în valoarea probabilității primare a acestei propoziții, lucru care permite recalcularea acestei probabilități cu ajutorul teoremei lui Bayes, sub denumirea de "*probabilitate posterioară*". Cu alte cuvinte, probabilitatea ca propoziția "$\mathbf{z_i}$" să fie adevărată când se ştie că evidenta "**E**" este adevărată se va nota cu "$\mathbf{P(z_i|E)}$".

In această teoremă este implicată şi o altă noțiune şi anume cea de "*probabilitate condiționată*" sau "*condițional*", care exprimă încrederea individuală că evidența "**E**" este confirmată atunci când propoziția "$\mathbf{z_i}$" este adevărată şi se notează cu "$\mathbf{P(E|z_i)}$". Pentru un set

de "n" propoziții, în condițiile existenței unei evidențe "**E**", probabilitățile posterioare pot fi calculate cu relația (5.2-1):

$$P(z_i \mid E) = \frac{P(E \mid z_i) \cdot P(z_i)}{\sum_{k=1}^{n} P(E \mid z_k) \cdot P(z_k)}$$

(5.2-1)

Valorile astfel obținute sunt valori normalizate, adică suma lor va fi 1.

Dacă sistemul de evidențe are mai multe componente, de pildă "m", atunci probabilitățile posterioare ale unui eveniment oarecare se vor calcula cu relația (5.2-2):

$$P(z_i \mid E_1 E_2 \ldots E_m) = \frac{\left[\prod_{j=1}^{m} P(E_j \mid z_i)\right] \cdot P(z_i)}{\sum_{k=1}^{n}\left[\prod_{j=1}^{m} P(E_j \mid z_k)\right] \cdot P(z_k)}$$

(5.2-2)

Pentru cazul mai multor propoziții şi o singură evidență se mai poate folosi şi noțiunea de "*probabilitate primară propagată*", calculabilă cu relația (5.2-3) pentru o anumită propoziție "z_i" :

$$P_p(z_i \mid E) = \frac{P(E \mid z_i) \cdot P(z_i)}{P(E \mid z_i) \cdot P(z_i) + P(E \mid \overline{z}_i) \cdot P(\overline{z}_i)}$$

(5.2-3)

Deşi aceste valori nu mai sunt normalizate, ele pot constitui valori primare pentru un nou calcul de valori posterioare, în prezența unei alte evidențe, obținându-se astfel din aproape în aproape influența fiecărei evidențe asupra valorilor posterioare ale probabilităților. Termenul suplimentar care apare la această ecuație sub forma "$P(E \mid \overline{z}_i)$" reprezintă nivelul individual de încredere că evidența "E" va fi valabilă şi în cazul când propoziția "z_i" nu este adevărată. Dacă evenimentul studiat implică mai multe evidențe, atunci relația (5.2-3) se poate scrie pentru fiecare evidență în parte în felul următor:

$$P_p(z_i \mid E_j) = \frac{P(E_j \mid z_i) \cdot P(z_i)}{P(E_j \mid z_i) \cdot P(z_i) + P(E_j \mid \overline{z}_i) \cdot [1 - P(z_i)]}$$

(5.2-4)

Dacă se intenționează studierea influenței unui sistem de evidențe format din "**k**" elemente asupra unui set de propoziții, atunci probabilitățile posterioare se vor determina cu relația (5.2-5):

$$P_p(z_i \mid E_1E_2...E_k) = \frac{P(E_k \mid z_i) \cdot P_p(z_i \mid E_1E_2...E_{k-1})}{P(E_k \mid z_i) \cdot P_p(z_i \mid E_1E_2...E_{k-1}) + P(E_k \mid \bar{z}_i) \cdot [1 - P_p(z_i \mid E_1E_2...E_{k-1})]}$$

(5.2-5)

Determinarea distribuțiilor primare "$\mathbf{P(z_i)}$" se face pe baza informațiilor existente asupra unei propoziții "$\mathbf{z_i}$", de natură subiectivă, obiectivă, sau ambele. Dacă aceste informații nu sunt suficiente pentru a defini valorile primare, atunci se poate presupune că toate propozițiile au aceeaşi valoare de adevăr, deci aceeaşi probabilitate, fapt care duce la o formă simplificată a relației anterioare:

$$P_a(z_i \mid E_1E_2...E_m) = \frac{\left[\prod_{j=1}^{m} P(E_j \mid z_i)\right]}{\sum_{k=1}^{n}\left[\prod_{j=1}^{m} P(E_j \mid z_k)\right]}$$

(5.2-6)

In cazul în care condiționalii "$\mathbf{P(E_k|z_i)}$" au valori particulare (0; 0,5; 1), distribuția primară se va face într-un mod special. De pildă dacă $P(E_k|z_i)=0$, evidența E_k nu se manifestă chiar dacă se ştie că propoziția "$\mathbf{z_i}$" este adevărată, în timp ce dacă $P(E_k|z_i)=1$ se poate spune că evidența $\mathbf{E_k}$ se manifestă cu certitudine dacă propoziția "$\mathbf{z_i}$" este adevărată. Cele două situații pot fi descrise şi prin formulările "*dacă z_i atunci non E_k*" respectiv "*dacă z_i atunci E_k*". Dacă $P(E_k|z_i)=0,5$, evidența nu mai are nici o relevanță, deci ea nu va mai afecta valorile probabilităților posterioare. Această valoare a condiționalului este folosită atunci când nu avem informații suficiente pentru a preciza alte valori. Exemplul următor încearcă să ilustreze acest caz.

5.2.b Exemplu de utilizare a relațiilor de calcul a probabilităților propagate

Se presupune că un dispozitiv de transfer într-un sistem flexibil de fabricație primeşte simultan trei cereri de deplasare către trei posturi diferite de lucru din sistem. El va deservi în mod evident una singură din aceste direcții, pe care trebuie să o aleagă pe baza unui sistem de criterii. Cele trei *propoziții* "$\mathbf{z_i}$" cu valori diferite de adevăr (opțional subiectiv sau obiectiv), sunt următoarele:

- sistemul de transfer alege ruta 1 pentru descărcarea unei maşini importante;
- sistemul de transfer alege ruta 2 pentru evacuarea unui produs finit;
- sistemul de transfer alege ruta 3 pentru adăugarea unui semifabricat într-un şir de aşteptare.

Sistemul de *evidențe* care acționează în acest caz poate cuprinde următoarele afirmații:

- distanța este maximă;
- distanța este minimă;
- se eliberează o maşină foarte importantă;
- se asigură evacuarea din sistem a unui produs finit;
- ruta este cea mai simplă.

Pornind de la aceste afirmații se poate alcătui baza de cunoştințe corespunzătoare acestui caz, sintetizată prin valorile condiționalilor din Tabelul 5.2-1.

Tabelul 5.2-1

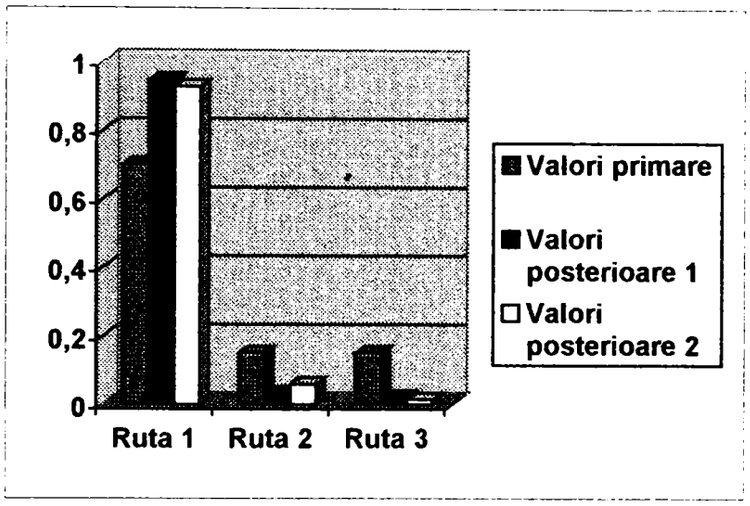
Propoziții	Ruta 1	Ruta 2	Ruta 3
Valori primare $P(z_i)$	0,7	0,15	0,15
distanța este maximă $P(E_1\|z_i)$	0,8	0,1	0,5
distanța este minimă $P(E_2\|z_i)$	0,2	0,8	0,5
se eliberează o maşină foarte importantă $P(E_3\|z_i)$	0,9	0,1	0,1
se asigură evacuarea din sistem a unui produs finit $P(E_4\|z_i)$	0,3	0,8	0,2
ruta este cea mai simplă $P(E_5\|z_i)$	0,8	0,9	0,7
Valori posterioare $P(z_i\|E_1E_2E_3E_4E_5)$ (conform (5.2-2))	0,9457*	0,03377	0,020205

Dacă se presupune că nu se cunoaşte influența alegerii rutei asupra distanței de parcurs şi a simplității traseului, valorile condiționalilor respectiv valorile posterioare vor fi conform cu datele din Tabelul 5.2-2.

Tabelul 5.2-2

Propoziții	Ruta 1	Ruta 2	Ruta 3
Valori primare $P(z_i)$	0,7	0,15	0,15
*distanța este maximă $P(E_1\|z_i)$	0,5	0,5	0,5
*distanța este minimă $P(E_2\|z_i)$	0,5	0,5	0,5
se eliberează o maşină foarte importantă $P(E_3\|z_i)$	0,9	0,1	0,1
se asigură evacuarea din sistem a unui produs finit $P(E_4\|z_i)$	0,3	0,8	0,2
*ruta este cea mai simplă $P(E_5\|z_i)$	0,5	0,5	0,5
Valori posterioare $P(z_i\|E_1E_2E_3E_4E_5)$ (conform (5.2-2))	0,92647*	0,058823	0,0147

Analizând relația de calcul a valorilor posterioare, se observă că toate valorile de "0,5" ale condiționalilor se simplifică şi deci nu influențează rezultatul. In ambele situații, ruta preferată va fi ruta numărul 1. Pe Graficul 5.2-1au fost reprezentate valorile posterioare în cele două cazuri de mai sus, când se ține cont de caracteristicile traseului ce urmează să fie parcurs (cazul 1) sau când nu se ține cont de aceste caracteristici (cazul 2). Se observă că ruta 1 este favorită în ambele situații, deci influența tipului de traseu ales este minimală în comparație cu ceilalți parametrii de decizie



Graficul 5.2-1

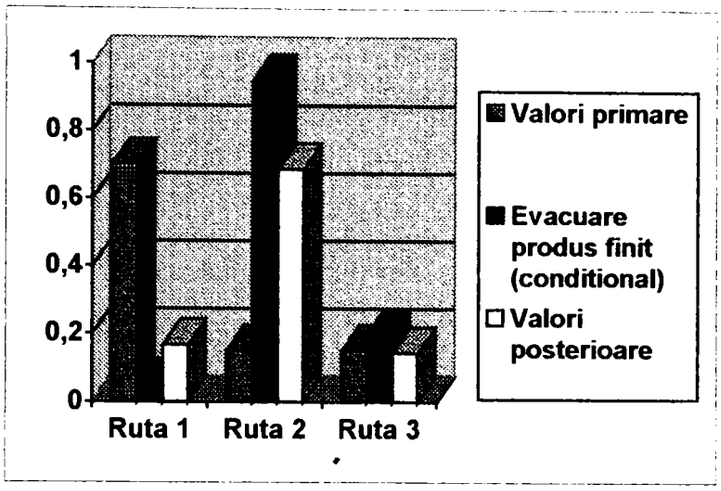
In continuare se presupune că este activă o singură evidență şi anume cea referitoare la evacuarea din sistem a unui produs finit. Baza de cunoştințe şi valorile condiționalilor vor fi ca în Tabelul 5.2-3.

Tabelul 5.2-3

Propoziții	Ruta 1	Ruta 2	Ruta 3
Valori primare $P(z_i)$	0,7	0,15	0,15
se asigură evacuarea din sistem a unui produs finit $P(E_4\|z_i)$	0,05	0,95	0,2
Valori posterioare $P(z_i\|E_4)$ (conform (5.2-2))	0,168	0,6867	0,14458

Se poate observa din analiza datelor ultimului tabel, că influența valorilor primare este foarte greu de contracarat, fiind nevoie de valori foarte nete ale condiționalilor pentru a putea reuşi să schimbăm o decizie luată pe baza valorilor primare, care nu țin cont de nici un fel de evidențe. Acest lucru devine vizibil şi pe Graficul 5.2-2, unde valoarea primară maximă care apare la ruta 1 (cu albastru) este transformată într-o valoare posterioară maximă pe ruta 2 (în

galben) cu ajutorul unui condițional foarte mare, aproape unitar pe această rută (în roşu), care exprimă faptul că pe ruta 2 se face evacuarea din sistem a unui produs finit.



Graficul 5.2-2

De cele mai multe ori, în calculele inginereşti, evidențele nu se manifestă doar cu valorile lor de "*adevărat*" sau "*fals*" ci cu o gamă mult mai largă de valori, care au un aspect stocastic. De aceea este necesară introducerea unui nou parametru "**C**", care se numeşte "*factor de incertitudine*" şi care exprimă încrederea individuală a observatorului sau a proiectantului că evidența este adevărată. Valorile acestui factor de incertitudine sunt cuprinse între 0 şi 1. Valoarea "0" indică faptul că evidența nu se va produce, iar valoarea 1 exprimă faptul că ea se va produce cu certitudine. Valoarea 0,5 nu are nici o semnificație, în sensul că evidența poate să se producă sau să nu se producă, cu probabilități egale. Vor fi deci posibile următoarele situații:

- evidența este adevărată, condiționalul va fi $P(E_i|z_j)$;
- evidența este falsă, condiționalul va fi 1- $P(E_i|z_j)$;
- evidența este total incertă, condiționalul va fi 0,5.

Aceste trei situații pot fi modelate cu ajutorul unei drepte în coordonate $[C_i;P(E_i,C_i|z_j)]$, care se va scrie în felul următor:

$$P(E_i,C_i \mid z_j) = (1-C_i)+(2\cdot C_i - 1)P(E_i \mid z_j)$$

(5.2-7)

Dacă se ține cont de aceste observații în ecuația (5.2-2), ea va avea forma (5.2-8):

$$P(z_i \mid E_1E_2...E_m) = \frac{\left[\prod_{j=1}^{m} P(E_j,C_j \mid z_i)\right]\cdot P(z_i)}{\sum_{k=1}^{n}\left[\prod_{j=1}^{m} P(E_j,C_j \mid z_k)\right]\cdot P(z_k)}$$

(5.2-8)

5.3. Decizia de rutare bazată pe utilizarea teoriei probabilităţilor subiective

Teoria probabilităţilor subiective prezentată pe scurt în cele de mai sus a fost folosită în cadrul tezei de doctorat în scopul modificării on-line a deciziilor de rutare pe baza unui sistem de criterii cărora li s-au asociat nişte valori ale condiţionalilor (coeficienţi de pondere) şi nişte valori ale probabilităţilor primare. Astfel, în situaţii de concurenţă, când programul de conducere a sistemului oferă mai multe posibilităţi de continuare a procesului, se apelează o rutină suplimentară denumită "subjective_probability", care va calcula probabilităţile posterioare ale tranziţiilor implicate la un moment dat şi va returna tranziţia cu valoarea cea mai mare a acestei probabilităţi. Aceasta va fi deci următoarea tranziţie lansată în execuţie în sistem.

Pentru a putea lua o decizie în ce priveşte tranziţia succesoare a unei situaţii curente s-a alcătuit un sistem de criterii de decizie sistematizate în Tabelul 5.3-4. Fireşte că aceste criterii pot fi rafinate în continuare pe baza experienţei acumulate în timp de către operatorii de sistem şi pot fi modificate pentru a se acomoda cât mai fidel cu nevoile reale din sistem. Criteriile de decizie prezentate în acest caz sunt doar un exemplu ilustrativ care să valideze metoda propriu-zisă. Aceleaşi observaţii sunt valabile şi în cazul valorilor de probabilitate primară şi ale coeficienţilor de pondere care apar în tabel.

Tabelul 5.3-4

Tranziţii / **Criterii de decizie**	**TriMF**	**TroMF**	**Frez**	**TriAGV**	**Masurare**	**TroMas**	**TriCloos**	**TroCloos**	**Sudare**	**TrDep1**	**TrDep2**	**Asmb**
Valori primare ($x10^{-2}$)	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3
Eliberarea frezei	0.7	0.9	0.8	0.6	0.6	0.4	0.5	0.5	0.5	0.5	0.5	0.6
Eliberarea robotului de sudare	0.3	0.3	0.5	0.5	0.5	0.5	0.7	0.9	0.8	0.5	0.5	0.5
Eliberarea AGV	0.7	0.5	0.5	0.2	0.5	0.5	0.7	0.5	0.5	0.7	0.7	0.5
introducerea unui nou reper în sistem	0.6	0.7	0.9	0.9	0.8	0.5	0.7	0.8	0.8	0.7	0.6	0.5

Pe baza datelor din tabel, care sunt preluate într-o structură C pentru a putea fi ulterior folosite în programul de decizie, se vor calcula valorile posterioare ale probabilităţilor subiective, care vor sta efectiv la baza luării deciziei. Coeficienţii de pondere pot avea în principiu orice valoare cuprinsă între 0 şi 1. Spre deosebire de aceştia însă, valorile

probabilităţilor primare trebuie să aibă suma unitară pentru tot setul de tranziţii studiat. Din acest motiv se impune o operaţie de normare a valorilor primare pentru setul de tranziţii în discuţie, astfel încât suma probabilităţilor lor să fie 1. In continuare se vor calcula valorile posterioare ale probabilităţilor pe baza relaţiei (5.2-2). Fragmentul din fişierul sursă care cuprinde funcţia "subjective_probability" este prezentat în Anexa 5-1.

In continuare este prezentat un exemplu pentru modul de operare al programului. Să presupunem că pe baza programului "control" care conduce propriu-zis sistemul de fabricaţie s-a ajuns în nodul "28" care are trei succesori:

- nodul "31" cu tranziţia "frezare'Frez";
- nodul "32" cu tranziţia "sudare'TroCloos";
- nodul "33" cu tranziţia "conveior'TriAGV".

Aceste valori au fost extrase din fişierul "succ_arce_trz" care este folosit ca parametru de intrare al programului "control" şi este prezentat în anexele de la capitolul 4. Pe baza acestor valori, programul "subjective_probability" va extrage din Tabelul 5.3-4 valorile corespunzătoare ale indicilor de probabilitate primară şi ale condiţionalilor, valori care au fost sistematizate în Tabelul 5.3-5, şi va calcula valoarea probabilităţilor posterioare care au fost redate pe ultima linie din tabel. Pe baza acestor valori, decizia multicriterială privind continuarea prelucrării în sistem va fi "frezare'Frez", ceea ce înseamnă ca se va transfera controlul în sistem către controlerul de la celula de frezare care va declanşa operaţia de frezare. Această decizie poate fi validată sau nu de către operator. In cazul nevalidării primei oferte se trece la oferta a doua şi aşa mai departe, până la epuizarea întregii liste.

Tabelul 5.3-5

Tranziţii / **Criterii de decizie**	**Frez**	**TriAGV**	**TroCloos**
Valori primare ($x10^{-2}$) $P(z_i)$ (valori normate)	33	33	33
Eliberarea frezei $P(E_1\|z_i)$	0.8	0.6	0.5
Eliberarea robotului de sudare $P(E_2\|z_i)$	0.5	0.5	0.9
Eliberarea AGV $P(E_3\|z_i)$	0.5	0.2	0.5
Încărcarea unui nou reper în sistem $P(E_4\|z_i)$	0.9	0.9	0.8
Valori posterioare ($P(z_i\|E_1E_2E_3E_4)$	0.44	0.13	0.43

Folosind relația (5.2-3) se pot determina şi alte genuri de influențe ale evidențelor acceptate asupra probabilităților primare, obținându-se aşa-numitele *"probabilități primare propagate"*. Se poate determina astfel de pildă influența pe care o are procesul de frezare sau de sudare asupra probabilităților primare. Rezultatele unui astfel de calcul sunt cuprinse în Tabelul 5.3-6. Acestea pot fi folosite ca noi valori ale probabilităților primare, dacă se consideră relevant, mai ales în situația în care este greu de determinat inițial care sunt probabilitățile obiective de declanşare a fiecărei tranziții.

Tabelul 5.3-6

Tranziții / **Criterii de decizie**	**TriMF**	**TroMF**	**Frez**	**TriAGV**	**Masurare**	**TroMas**	**TriCloos**	**TroCloos**	**Sudare**	**TrDep1**	**TrDep2**	**Asmb**
Valori primare ($x10^{-2}$)	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3
Eliberarea frezei (evidența 1)	0.7	0.9	0.8	0.6	0.6	0.4	0.5	0.5	0.5	0.5	0.5	0.6
Eliberarea robotului de sudare (evidența 2)	0.3	0.3	0.5	0.5	0.5	0.5	0.7	0.9	0.8	0.5	0.5	0.5
Valori primare ($x10^{-2}$) pe baza evidenței 1 (5.2-3)	10	12	11	9.1	9.1	5.7	6.8	6.8	6.8	6.8	6.8	9.1
Valori primare ($x10^{-2}$) pe baza evidenței 2 (5.2-3)	5.2	5.2	6.8	6.8	6.8	6.8	13	15	14	6.8	6.8	6.8

Valorile obținute prin aplicarea formulei (5.2-3) au fost normate astfel încât să poată fi folosite ca probabilități primare pentru un proces ulterior de decizie.

In momentul în care se cunoaşte decizia sistemului pentru una din tranzițiile posibile, echipamentul de comandă (calculatorul central) va da un ordin de execuție la una din celulele independente ale sistemului prezentate în capitolul anterior. Aceasta va lansa în execuție propriul program de lucru (care nu face obiectul prezentei lucrări) şi va prelua comanda integrală a celulei până la terminarea operațiilor respective. La sfârşitul lucrului controlerul local va furniza un semnal de sfârşit de proces, care va declanşa la nivelul computerului central o nouă secvența de decizie în legătură cu etapa următoare în desfăşurarea procesului.

Tot acest comportament al sistemului de conducere legat de alegerea unei căi de rutare optime pe baza unui sistem de condiționali este condus cu ajutorul funcției *"subjective_probability"* care este prezentată în anexele următoare. Pe lângă această funcție sistemul mai are nevoie şi de altele care cooperează cu prima. Astfel funcția

"read_subj_prob_val" are rolul de a prelua dintr-un fişier text un stream de date care conţin valorile indicilor de probabilitate pentru valorile primare şi condiţionalii luaţi în considerare la un moment dat. Acest fişier se numeşte "subjective" şi el poate fi editat cu orice editor de text fără nici un fel de probleme speciale. In acest fel programul devine independent de setul iniţial de date. Valorile citite din acest fişier vor fi încărcate într-o variabilă structurată de tipul *"float m_values[5][12]"* care conţine un prim indice pentru setul de date şi un al doilea pentru valorile concrete ale probabilităţilor (vezi şi structura acestei variabile în Anexa 5-2). Fişierul cu valorile probabilităţilor folosit la încercări este prezentat în Anexa 5-3. Pentru citirea linilor din fişier se foloseşte funcţia C "fscanf".

După citirea valorilor din fişierul sus-menţionat şi încărcarea lor în variabila structurată despre care s-a vorbit, aceste valori pot fi folosite pentru luarea deciziilor de rutare în sistem. Astfel, dacă funcţia *"getsucc"* returnează mai mult de un succesor pentru un nod, în fereastra de dialog cu operatorul va apare oferta completă de tranziţii posibile, determinate cu funcţia *"gettranz"*. Intre aceste tranziţii operatorul este invitat să aleagă una, a cărei declanşare va determina starea următoare a sistemului corespunzătoare unui nou nod. Dacă doreşte, operatorul poate selecta opţiunea "0", care va declanşa "serviciile" funcţiei *"subjective_probability"*. In urma rulării acestei funcţii se va returna una din variantele oferite anterior, care s-a calificat pe baza sistemului de condiţionali formulat iniţial. Operatorul poate să accepte această variantă sau să o respingă, caz în care i se oferă din nou meniul anterior de selecţie, în care îşi poate exprima o nouă opţiune. O astfel de fereastră de dialog este prezentată în Figura 5.3-1.

Dacă operatorul optează pentru alegerea rutei optime pe baza probabilităţilor subiective, în program se va declanşa mai întâi o rutină de calcul a probabilităţilor posterioare cu ajutorul relaţiei (5.2-2). Valorile probabilităţilor corespunzătoare tranziţiilor cu care se vor efectua calculele se vor identifica din structura în care au fost încărcate, cu ajutorul funcţiei *"GetCmdIDFromTrz(trz_name)"*. Urmează apoi calculul propriu-zis al valorilor probabilităţilor posterioare. Pentru cea mai mare dintre aceste valori se va returna numele tranziţiei şi numărul nodului succesor aşa cum rezultă el din lista de succesori care constituie parametru de intrare al programului "control".

Procedura de optimizare a rutelor de transfer pe baza probabilităţilor subiective se înscrie printre metodele de control a fluxurilor de materiale după criterii care să îmbunătăţească performanţele sistemului de fabricaţie. Funcţia obiectiv a optimizării o constituie reducerea riscului de apariţie a blocajelor în sistem. Aşa cum s-a arătat pe larg în capitolul I, unde se vorbeşte despre sisteme de fabricaţie, fluxurile de materiale sunt

inseparabile de fluxurile informaţionale. **Din acest motiv se poate afirma că odată cu optimizarea circulaţiei materialelor s-a realizat şi o optimizare a fluxului de informaţie care o însoţeşte**. De asemenea, prin propunerile de modificare a arhitecturii sistemului de comandă s-a adus iarăşi un aport la îmbunătăţirea fluxurilor informaţionale. Programele de calcul care au fost prezentate şi care realizează o strategie de transfer optimizată sunt ele însele dovezi ale coexistenţei inseparabile a aspectelor materiale şi a celor informaţionale în desfăşurarea ciclului de fabricaţie dintr-un sistem de fabricaţie flexibilă.

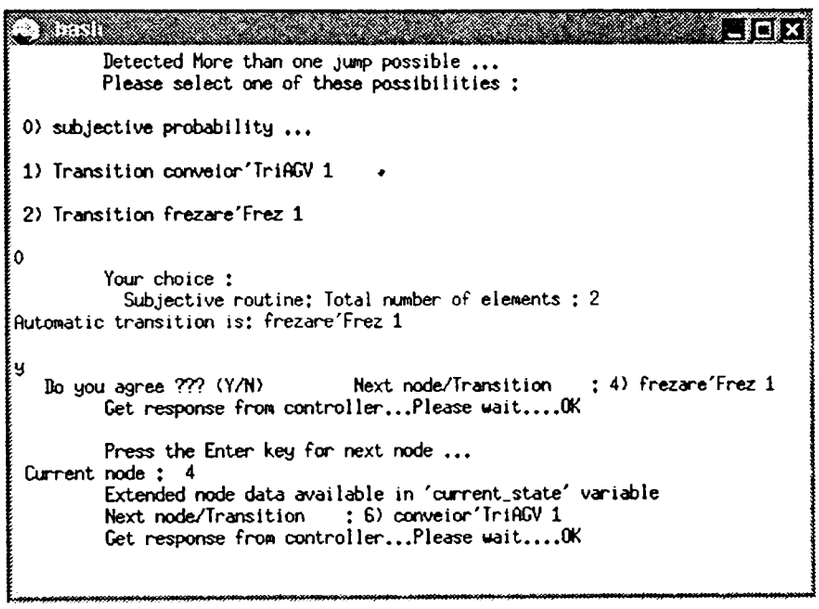


Figura 5.3-1

5.4. Anexe

Anexa 5-1 Fragment din fişierul sursă C care cuprinde funcţia "subjective probability" folosită pentru adoptarea automată a deciziilor de rutare pe baza teoriei probabilităţilor subiective

```
void read_subj_prob_val(void)

{

     char temp[55], nbr_temp[10];

     int i;

     for (i = 0; i < 5; i++)

     {
          fscanf(subj_prob, "%s", temp);

          fscanf(subj_prob, "%s %s", temp, nbr_temp);

          m_values[i][TriMF] = atof(nbr_temp);

          fscanf(subj_prob, "%s %s", temp, nbr_temp);

          m_values[i][TroMF] = atof(nbr_temp);

          fscanf(subj_prob, "%s %s", temp, nbr_temp);

          m_values[i][Frez] = atof(nbr_temp);

          fscanf(subj_prob, "%s %s", temp, nbr_temp);

          m_values[i][TriAGV] = atof(nbr_temp);

          fscanf(subj_prob, "%s %s", temp, nbr_temp);

          m_values[i][Masurare] = atof(nbr_temp);

          fscanf(subj_prob, "%s %s", temp, nbr_temp);

          m_values[i][TroMas] = atof(nbr_temp);

          fscanf(subj_prob, "%s %s", temp, nbr_temp);

          m_values[i][TriCloos] = atof(nbr_temp);

          fscanf(subj_prob, "%s %s", temp, nbr_temp);
```

```
            m_values[i][TroCloos] = atof(nbr_temp);
            fscanf(subj_prob, "%s %s", temp, nbr_temp);
            m_values[i][Sudare] = atof(nbr_temp);
            fscanf(subj_prob, "%s %s", temp, nbr_temp);
            m_values[i][TriDep1] = atof(nbr_temp);
            fscanf(subj_prob, "%s %s", temp, nbr_temp);
            m_values[i][TriDep2] = atof(nbr_temp);
            fscanf(subj_prob, "%s %s", temp, nbr_temp);
            m_values[i][Asmb] = atof(nbr_temp);
      }
}

/*get command id from tranzition string*/

int GetCmdIDFromTrz(char* tranz)
{
      char command[20];
      int       j, count;
      for (j = 0; j < strlen(tranz); j++)
            if (tranz[j] == 0x27)
                  {j++;break;}
      count = 0;
      while (tranz[j] != ' ' && j < strlen(tranz))
      {
            command[count] = tranz[j];
            count++;j++;
      }
      command[count] = '\0';
```

```
    if (!strcmp(command, "TriMF"))     {return 0;}

    if (!strcmp(command, "TroMF"))     {return 1;}

    if (!strcmp(command, "Frez"))      {return 2;}

    if (!strcmp(command, "TriAGV"))    {return 3;}

    if (!strcmp(command, "Masurare")) {return 4;}

    if (!strcmp(command, "TroMas"))    {return 5;}

    if (!strcmp(command, "TriCloos")) {return 6;}

    if (!strcmp(command, "TroCloos")) {return 7;}

    if (!strcmp(command, "Sudare"))    {return 8;}

    if (!strcmp(command, "TriDep1"))   {return 9;}

    if (!strcmp(command, "TriDep2"))   {return 10;}

    if (!strcmp(command, "Asmb"))      {return 11;}

    return -1;
}

int subjective_probability(int n,int *nodes)

{

    int        i, j, k, index_maxim;
       char* tmp;

    float      val_primare, val_pos[12], maxim = 0;

    float   prod_sus = 1.0, prod_jos = 1.0, suma_jos = 0;

    int        cmdid[12];

    val_primare = 100 / (float)n;

    for (j = 0; j < 12; j++)

        m_values[Start_Values][j] = val_primare;
    printf("\n            Subjective routine: Total number of
elements : %d",n);

    for(i=0; i < n; i++)
       {
```

```
            tmp = gettranz(nodes[i], trz);
        cmdid[i] = GetCmdIDFromTrz(tmp);
            if (tmp) free(tmp);
    }
for (i = 0; i < n; i++)

{

    prod_sus = prod_jos = 1;

    suma_jos = 0;

    for (j = 0; j < 5; j++)

        prod_sus *= m_values[j][cmdid[i]];

    for (k = 0; k < n; k++)

    {

        prod_jos = 1;

        for (j = 0; j < 5; j++)

            prod_jos *= m_values[j][cmdid[k]];

        suma_jos += prod_jos;

    }

    val_pos[i] = prod_sus / suma_jos;

}

for (i = 0; i < n; i++)

    if (val_pos[i] > maxim)

    {

        maxim = val_pos[i];

        index_maxim = nodes[i];

    }


return(index_maxim);
}
```

Anexa 5-2 Fişierul antet "subject.h" care cuprinde structura de date folosită pentru înmagazinarea valorilor condiţionalilor şi a probabilităţilor primare.

```
#define bool unsigned char

#define Start_Values     0
#define Free_Mill        1
#define Free_Cloos       2
#define Free_AGV         3
#define New_PartIn       4



#define    TriMF         0
#define    TroMF         1
#define    Frez          2
#define    TriAGV        3
#define    Masurare      4
#define    TroMas        5
#define    TriCloos      6
#define    TroCloos      7
#define    Sudare        8
#define    TriDep1       9
#define    TriDep2       10
#define    Asmb          11
```

Anexa 5-3 Fişierul text "subjective" care cuprinde valorile indicilor de probabilitate precum şi probabilităţile primare pentru rutarea dinamică.

```
Start_Values
TriMF      0.08333
TroMF      0.08333
Frez       0.08333
TriAGV     0.08333
Masurare   0.08333
TroMas     0.08333
TriCloos   0.08333
TroCloos   0.08333
Sudare     0.08333
TriDep1    0.08333
TriDep2    0.08333
Asmb       0.08333
Free_Mill
TriMF      0.7
TroMF      0.9
Frez       0.8
TriAGV     0.6
Masurare   0.6
TroMas     0.4
TriCloos   0.5
TroCloos   0.5
Sudare     0.5
TriDep     0.5
TriDep2    0.5
Asmb       0.6
Free_Cloos
TriMF      0.3
TroMF      0.3
Frez       0.5
TriAGV     0.5
Masurare   0.5
TroMas     0.5
TriCloos   0.7
TroCloos   0.9
Sudare     0.8
TriDep1    0.5
TriDep2    0.5
Asmb       0.5
Free_AGV
TriMF      0.7
TroMF      0.5
Frez       0.5
TriAGV     0.2
Masurare   0.5
TroMas     0.5
TriCloos   0.7
```

```
TroCloos   0.5
Sudare     0.5
TriDep1    0.7
TriDep2    0.7
Asmb       0.5
New_PartIn
TriMF      0.6
TroMF      0.7
Frez       0.9
TriAGV     0.9
Masurare   0.8
TroMas     0.5
TriCloos   0.7
TroCloos   0.8
Sudare     0.8
TriDep1    0.7
TriDep2    0.7
Asmb       0.5
```

5.5. Bibliografie

1. Crişan, I., Drăgănoiu, Gh., Predoi, A., 1988, Sisteme flexibile de montaj cu roboţi şi manipulatoare. Editura Tehnică, Bucureşti.
2. Dreucean, M., 1994, Unele aspecte ale dependenţei de context a reţelelor neuronale. Simpozionul national de roboţi industriali, Timişoara , 1994.
3. Dreucean, M., 1997, Exemplu de modelare cu reţele Petri a unei celule flexibile de fabricaţie. Sesiunea de comunicări ştiinţifice, Ediţia a IV-a, Arad, 1997.
4. Dreucean, M., 1997, Modelarea fluxurilor de materiale şi de informaţie în sisteme de fabricaţie flexibile, Referatul nr. 2, Timişoara.
5. Dreucean, M., 1998, Modeling Flexible Manufacturing Systems Using Subjective Probabilities, 2nd International Symposium of Industrial Engineering SIE '98, Belgrad, Yugoslavia.
6. Hamid R. Parsaei, William G. Sullivan, 1993, Concurrent Engineering. Chapman & Hall, Glasgow, New-York, Tokyo, Melbourne, Madras.
7. Hietiko, E., 1996, Computer Aided System for Preliminary Design of Screen Cylinder Variants, C Technica 90, Acta Universitatis Ouluensis Technica, Departement of Mechanical Engineering, Oulu, Finland.
8. Steven Y. Liang, Tsu-Chin Tsao (ASME), 1990, Monitoring and Control for Manufacturing Processes. The American Society of Mechanical Engineers, New-York.

6. CONTRIBUȚII PERSONALE ŞI CONCLUZII

6.1. Cuprins

6.2. Contribuții personale

6.2.a In domeniul teoretic

- sinteza unui volum vast de cunoştinţe din domeniul sistemelor de fabricaţie;
- sistematizarea parametrilor cantitativi pentru aprecierea performanţelor unui sistem de fabricaţie;
- prezentarea conceptului de celulă de fabricaţie independentă;
- aplicaţii ale teoriei informaţiei la studiul conţinutului informaţional al unor operaţii tehnologice;
- precizări în legătură cu definirea noţiunilor de "modelare" şi "simulare";
- definirea locului modelării şi simulării în proiectarea sistemelor de fabricaţie;
- definirea locului sistemelor de fabricaţie în cadrul conceptului de "inginerie concurentă";
- prezentarea analitică a principalelor metode de simulare pentru sisteme cu evenimente discrete;
- prezentarea analitică a principalelor produse Software pentru simularea sistemelor cu evenimente discrete;
- identificarea cauzelor apariţiei blocajelor în sistem şi elaborarea unei strategii de eliminare a lor pe baza logicii de rutare;
- descrierea unei metode de conducere a sistemului de fabricaţie pe baza strategiei originale de evitare a blocajelor;
- definirea conceptului de "magistrală de blocaj" şi elaborarea metodelor de determinare a acestor magistrale pentru un sistem dat;
- utilizarea conceptului de "probabilitate subiectivă" pentru alegerea succesiunii optimale a fazelor ciclului de fabricaţie dintr-o celulă flexibilă.

6.2.b In domeniul practic

- realizarea modelului cu reţele Petri a sistemului de fabricaţie existent în laboratorul Catedrei OMM;
- implementarea a două variante ale logicii de rutare în modelul construit pe baza sistemului de fabricaţie flexibilă al Catedrei OMM;

- contribuţii la concepţia generală a fluxului tehnologic în sistemul de fabricaţie flexibilă din Laboratorul de Sisteme de Fabricaţie Flexibilă al Catedrei OMM a Facultăţii de Mecanică;
- contribuţii la concepţia subsistemului de transfer a materialelor din componenţa sistemului de fabricaţie existent în laboratorul Catedrei OMM;
- elaborarea a două variante de rutare a pieselor în sistemul flexibil de fabricaţie;
- definirea principalilor parametrii cantitativi ai funcţionării sistemului pe baza rezultatelor simulării;
- analiza critică a valorilor parametrilor cantitativi ai funcţionării sistemului pentru cele două variante ale logicii de rutare;
- scrierea şi dezvoltarea unui pachet de funcţii în limbaj CpnML pentru investigarea sistemului din punct de vedere cantitativ;
- scrierea şi dezvoltarea unui sistem de funcţii în limbaj CpnML pentru analiza grafului de incidenţă în scopul determinării condiţiilor de evitare a blocajelor;
- scrierea unor funcţii în limbaj CpnML pentru salvarea în fişiere a descriptorilor de noduri generaţi de graful de incidenţă;
- scrierea programului C "getdata" pentru constituirea bazelor de date ale descriptorilor de noduri ai sistemului;
- scrierea programului C "control" pentru conducerea sistemului în scopul evitării blocajelor;
- scrierea funcţiei C “subjective_probability” şi a funcţiilor conexe pentru implementarea în model a conceptului de “probabilitate subiectivă”;
- efectuarea unor teste de conducere a celulei de frezare a sistemului de fabricaţie cu ajutorul programului "control" bazat pe metoda de evitare a blocajelor propusă de autor;
- realizarea unei arhitecturi a sistemului de conducere bazată pe împărţirea sistemului de fabricaţie în mai multe celule cu sarcini de lucru independente;

6.3. Concluzii

La debutul lucrării de doctorat cu titlul “Optimizarea fluxului de materiale şi informaţii în sisteme de fabricaţie flexibilă robotizate” s-au propus spre rezolvare două probleme:

- analiza comparată, pe baza unor indici cantitativi, a diverselor strategii de control a fluxului de materiale (strategii de rutare sau strategii de transfer, optimizarea fluxurilor materiale);
- conducerea sistemului în scopul evitării blocajelor (optimizarea fluxurilor informaționale).

Lucrarea prezintă o *analiză comparativă a două strategii de rutare*, prima bazată strict pe partajarea resurselor din sistem şi a doua bazată pe tehnologia de execuție a pieselor. Ambele strategii de rutare au fost implementate la nivelul a două modele care au fost apoi folosite pentru simulare cu ajutorul rețelelor Petri colorate.

Analiza indicatorilor cantitativi folosiți evidențiază avantajele celei de-a doua variante de transfer, bazată pe tehnologia de execuție a pieselor, față de prima variantă, bazată strict pe partajarea resurslor.

Conducerea sistemului în scopul evitării blocajelor se bazează pe o implementare originală a ideii de utilizare a grafului de incidență rezultat în urma simulării. Pe baza unui set de funcții dezvoltate de autor, se poate obține din graful de incidență o bază de date externă mediului de dezvoltare a modelului, care conține toate stările posibile ale sistemului precum şi legăturile dintre ele. *Aceste legături implementează logica de rutare din model.* Se poate obține astfel un program de conducere cu logică variabilă atâta timp cât acest program nu face decât să determine trecerea sistemului dintr-o stare în alta pe baza relațiilor din model. **Este pe deplin justificat astfel faptul că toate eforturile de optimizare se vor canaliza spre model şi spre logica de transfer a materialelor pe care acesta o include**. După ce aceste aspecte au fost clarificate, **succesiunea de stări (noduri) ale sistemului dedusă din simulare se va transpune în practică pe sistemul real cu ajutorul programelor dezvoltate de autor**.

Pentru situațiile de conflict sau de concurență, când un nod are mai mulți succesori posibili, s-a realizat un program de selecție a rutei optime pe baza teoriei "probabilităților subiective" şi a conceptului de "magistrală de blocaj".

Metoda a fost verificată practic pentru celula de frezare din cadrul sistemului de fabricație instalat în laboratorul Catedrei OMM.

Se poate deci concluziona că cele două obiective propuse inițial au fost îndeplinite, obținându-se atât o metodă de apreciere a performanțelor unui sistem de fabricație bazată pe rezultatele simulării, cât şi **un program de conducere adaptabil oricărui tip de sistem de fabricație**, cu condiția ca acest sistem să fie modelat în prealabil cu ajutorul programului DesignCPN pentru a putea folosi funcțiile speciale care conduc la construcția bazei de date a stărilor posibile ale sistemului. Deşi teoria prezentată a fost grupată în jurul sistemului de

fabricație real aflat în laboratorul Catedrei de Organe de Mașini și Mecanisme, algoritmii și programele prezentate în lucrare pot fi cu ușurință adaptate oricărui sistem de fabricație, în condițiile prezentate mai sus. Se poate afirma că procedeele de analiză prezentate în lucrare se constituie într-o metodă de validare puțin costisitoare a sistemelor de fabricație încă din faza de proiectare.

Considerentele de ordin material au fost cele care au împiedicat validarea experimentală pe scară mai largă a programului de conducere. Cu toate acestea, funcționarea fiecărei celule din sistem fiind identică din punct de vedere informațional cu cea a celulei de frezare, se poate afirma că metoda de conducere propusă are reale șanse de a fi utilizată la conducerea întregului sistem. Un alt motiv pentru restrângerea ariei experimentale l-a constituit lipsa programelor de control local a celulelor independente, aceste programe făcând obiectul unor altor lucrări de cercetare sau de doctorat care sunt în derulare.

Lucrarea de doctorat prezentată **deschide perspectiva realizării unor sisteme de conducere bazate pe modelare-simulare**, sisteme a căror funcționare nu poate să ocolească stările determinate pe baza simulării. Această concepție a sistemului de conducere este în domeniul controlerelor de sistem ceea ce este conducerea "naster-slave" în domeniul roboților. Sistemul "master" în acest caz este modelul iar "sclavul" este sistemul real.

Continuarea cercetărilor pe linia deschisă de această lucrare ar trebui să se orienteze în primul rând spre aplicarea unor procedee de comunicare în rețea Ethernet sau MAP de tip "client-server" dintre controlerul central și cele locale. De asemenea trebuiesc continuate programele de conducere locală pe baza unei concepții unitare, care să permită preluarea din sistem a parametrilor ce definesc starea momentană ("current_state") și compararea lor cu baza de date a stărilor posibile ale sistemului.

Timișoara, la 24 aprilie 1999