Simularea Visuală a Arhitecturilor de Calcul Multiagent în Petri Nets Explorer

Alexei CORDUNEANU
Universitatea Tehnică a Moldovei
alexei.corduneanu@yahoo.com

***Abstract* — Acest articol prezintă conceptele de bază ale simulării arhitecturilor de calcul multiagen utilizînd mediul de modelare şi simulare vizuală Petri Nets Explorer. Acest sistem software dispune de o interfață grafică intuitivă pentru a crea o varietate de elemente ale sistemelor de calcul multiagent. La fel ne furnizează un mediu de simulare a modelelor arhitecturilor create pentru a uşura lucrul de cercetare în domeniu.**

***Index Terms* — modelare, multiagent, Petri Nets, membrane.**

I. INTRODUCERE

Sistemele multiagent sunt un subdomeniu al ştiinței calculatoarelor relativ nou, apărut în anii 1980 şi au obținut o largă recunoştință către mijlocul anilor 1990. Interesul sporit față de sistemele multigent a fost stimulat de părerea că agenții sunt o paradigmă software care este în stare să reprezinte sistemele distribuite de scară largă şi extra larga aşa cum ar fi Internet-ul.[1,5,8] Sistemele multiagent au devenit o metaforă naturală pentru înțelegerea construcția sistemelor sociale. Ideia de sistem multiagent nu ține doar de un anumit domeniu ci îşi gaseşte aplicabilitatea în diferite domenii.

Sistemele multiagent reprezintă nişte sisteme compuse din mai multe elemente ce interacționează numeite *agent.* Agenții sunt nişte sisteme de calcul cu două proprietați importante de bază care le descriu. În primul rînd, ei sunt în stare să acționeze de sine stătător – luînd decizia ce trebuie să îndeplinească ca să obțină scopul stabilit. În al doilea rînd agenții interacționează cu alți agenți – efectuînd un schimb de informație cu scopul de a stabili relația de: cooperare, coordonare, nogociere, etc. [1].

Una din metodele de studiere a sistemelor multiagent este modelarea lor. Unul din conceptele apropiate pentru acest studiu îl prezintă modelul de calcul membranal [10] . Unde fiecare membrană reprezintă un agent rațional şi interacțiunea acestuia cu alți agenți este descrisă de interacțiunea dintre membrane.

II. PROCESUL DE MODELARE

Dezvoltarea informaticii ca ştiință a fost stimulată de progresul tehnico-ştiințific în domeniul sistemelor de calcul şi tehnologiilor informaționale. Scopul primar al ştiinței fiind de a găsi noi metode de soluționare a problemelor de calcul computațional.[2]

Inițial sistemele de calcul s-au dezvoltat ca o ramură separata a matematicii aplicate. Şi primele probleme de care se preocupa erau pur calcule matematice, analiza seturilor complexe de date, etc. Actualmene însă, ştiința calculatoarelor este independentă, avînd metodele sale de cercetare (dar oricum bazate pe metodele matematice). Ştiința calculatoarelor se ocupa actual de soluționarea problemelor din diferite domenii, cum ar fi: fizica, chimia, biologia, economia, ecologia, sociologia, etc. Chiar orice industrie în toată lumea aplică ştiința calculatoarelor pentru a soluționa problemele bazate pe metodele matematice. [4] Acest fapt nu e întîmplător. Înainte de a soluționa o problemă este necesar de a descrie procesul domeniului printr-un concept matematic – *model*. Conceptele matematice ce pot sta la baza unui model sunt: funcții, ecuații, inecuații, sisteme de ecuații, etc.

De ce modelarea? În primul rînd furnizează un mod de interpretare inteligent al procesului din viața cotidiană. Multe ştiințe înca nu sunt bazate pe modele, deoarece modelele sunt un concept relativ nou. Diferite modele pot descrie orice proces. La fel modelele ne ajută să interpretăm volume mari de date, de a extrage informația şi cunoştințe din ele, altfel datele vor rămîne date. Şi în sfîrşit, dar nu mai puțin important modelele ne ajută să luăm decizii organizaționale şi strategice.[9]

Există mai multe instrumente şi concepte pentru modelarea sistemelor de calcul cum ar fi: GPSS, Lab VIEW, rețele Petri, algoritmi genetici, rețele neuronale, ş.a. [8] Fiecare instrumentar de modelare este dotat de o descriere matematică a modelului care permit modelarea a diferitor procese inclusiv şi din domeniul sistemelor de calcul. Însă în ultimii ani s-a demonstrat că sistemele de modelare bazate pe elemente de inteligență artificială sunt mult mai capabile în acest sens. Ele sunt în stare să dea o soluție simplă pentru probleme cu grad sporit de complexitate. Acest fapt ne sugerează utilizarea formalismelor bazate pe inteligența artificială pentru a descrie modelele din domeniul sistemelor de calcul.

III. ARHITECTURA ABSTRACTĂ A AGENTULUI INTELIGENT

Putem cu uşurință formaliza o viziune abstractă a unui agent. Inițial să considerăm că mediul poate sa se afle într-un set E finit de stări discrete.

De menționat faptul că pentru cazul agentului este indiferent dacă mediul este discret sau continuu, este doar o presupunere de modelare, care ne spune că orice sistem continuu poate fi descris de un set de stări discrete cu orice

nivel de acurateţe.

Agenţii sunt definiţi să posede un set de acţiuni disponibile, care transforma starea mediului. Fie:

Ac = {a, a',...} – un set finit de acţiuni.

Modelul de bază interacţionează cu mediul în felul următor. Mediul se află într-o stare iniţială şi agentul alege o acţiune corespunzătoare acestei stări a mediului. Ca rezultat al acestei acţiuni, mediul răspunde cu un set finit de stări posibile. Însă oricum mediul se va stabili în doar una din stări posibile, agentul activează în mediu parţial cunoscut deci nu ştie din timp în ce stare se va stabili mediul. Bazîndu-se pe starea a doua iar decide care acţiune sa fie îndeplinită. Mediul răspunde cu una din stările posibile, la care agentul iar alege care acţiune să fie îndeplinită şi aşa mai departe.

Drumul r, parcurs de agent într-un mediu este secvenţa de stări şi acţiuni respective:

r :e0

Fie

• R este setul tuturor secvenţelor posibile finite (peste mediul E şi acţiunile AC);

• RAC – este subsetul de stări care decurg la o acţiune; şi

• RE – este subsetul de secvenţe care finisează într-o anumită stare a mediului.

Vom utiliza r, r',... pentru a descrie membrii din R. Petru a reprezenta efectul acţiunilor agentului asupra stării mediului vom introduce funcţia de transformare

RAC → □(E).

Astfel starea este transformată la o secvenţă (se presupune că secvenţa finisează cu o acţiune a agentului), lo – un set de stări posibile ale mediului care pot rezulta în urma efectuării acţiunii.[11]

Două momente important de definit. Primul, mediul se presupune să fie dependent de istoria stărilor sale. În alte cuvinte mediul nu este doar dependent de acţiunile agentului ci şi de starea în care se află. Acţiunea executată anterior la fel joacă rolul său în determinarea stării curente. Într-al doilea rînd, această definiţie ne introduce indeterminismul medului, în ceia ce ţine de restul acţiunilor care vor fi executate în una din stări.

Daca T(r) = 0 (unde r este presupus să finiseze cu o acţiune), atunci nu există nici o stare posibilă ca succesoare în r. În cazul dat spunem ca sistemul şi-a sfîrşit drumul. Vom considera la fel că orice parcurs are sfîrşit.

Formulăm un mediu ca un triplet Env = (E, e0, □τ), unde E este un set de stări posibile ale mediului, e0 □ E este o stare iniţială, iar τ este funcţia de transformare a stării.

Acum trebuie sa definim modele de agenţi care populează mediul. Deci modelam agenţii ca fiind funcţii care definesc parcursul cu acţiunile:

Ag : RE → Ac.

Aşa dar, un agent ia decizia care acţiune să execute în baza istoriei sistemului, observatorul căreia fusese.[11]

De menţionat, că deoarece mediul este implicit nedeterminist, agenţii sunt presupuşi să fie determinişti. Fie Ag setul tuturor agenţilor.

Definim sistemul fiind un cuplu constituit din agent şi mediu. Orice sistem va avea un set de parcursuri posibile asociat cu aceste; notăm setul de parcurgeri al agentului Ag în mediul Env ca R(Ag, Env).[11] Pentru simplitate vom presupune că R(Ag, Env) conţine doar parcursuri finite, de exemplu, parcursul r aşa ca r nu are stări succesoare: T(r) = 0 (nu vom considera parcursuri infinite iniţial). Formal, avem secvenţa (e0, a0, e1, a1, e2, a2, …) reprezintă un parcurs al agentului Ag în mediul Env = (E, e0, T) dacă:

(1) e0 este starea iniţială a mediului Env.

(2) a0 = Ag(e0); şi:

(3) pentru u >0

eU □ τ(e0, a1, … aU-1),

unde

aU □ Ag(e0, a1, … aU).

Aceşti doi agenţi Ag1 şi Ag2 sunt comportamental echivalenţi în raport cu mediul Env dacă şi numai dacă R(Ag1,Env) = R(Ag2,Env), şi simplu comportamental echivalenţi dacă şi numai dacă sunt comportamental echivalenţi în raport cu orice mediu.

IV. REŢELE PETRI MEMBRANARE

Unul din instrumentele de modelare moderne este formalismul reţele Petri. Una din extensiunile acestuia fiind reţelele Petri membranale.

Calculul membranal este un domeniu nou, provocator şi în continua dezvoltare, situat la frontiera dintre ştiinţele sistemelor de calcul, informatice, matematice şi cele biologice. Cercetătorii din ştiinţa calculatoarelor şi informatică utilizează din ce în ce mai mult concepte şi modele inspirate din sistemele biologice precum şi simulările software derivate din aceste modele formale. P sistemele, menţionate şi ca fiind sisteme cu membrană, sunt o clasă de modele computaţionale paralel/distribuite inspirate din structura celulelor ţesutului viu [10]. Interesul P sistemelor înrudite cu modelul PN de calcul conduce la mai multe rezultate importante în probleme de decidabilitate, verificare şi simulare a proceselor de calcul. În această direcţie au fost efectuate unele eforturi pentru a simula P sistemele cu reţele Petri [10] pentru verificarea multor proprietăţi comportamentale folositoare, cum ar fi conexitatea, mărginirea, viabilitatea, reversibilitatea, lipsa de blocaje, etc.

În continuare, inspirând-ne de lucrările lui G. Ciobanu, G. Păun şi Nishida, în care sunt introduse şi cercetate diferite variante de P sisteme, ne propunem sa abordăm o nouă metodă de descriere ierarhică a proceselor de calcul, a exprimate prin componente de subreţele Petri membranale, de rescriere dinamică [10], folosind expresii descriptive (DE) [10]. Pentru a exprima localităţile, organizarea ierarhică şi comportamentul sistemelor de calcul mobile şi reconfigurabile, definim sistemel P hibride cu membrane active şi în baza cărora definim reţele Petri membranale stocastice, numite reţele-DMH, care pot modifica în mod dinamic (în dependenţă de marcajul curent), propria structură şi atributele sale prin reguli de rescriere a unor componentele DE.

În continuare, succint prezentăm numai modele de reţelele membranale, deoarece celelalte tipuri de modele reţele Petri pot fi obţinute în calitate de cazuri particulare. Mai detaliat se pot consulta lucrările recente ale autorului [3, 9, 10]. O descriere succintă a P sistemelor (sisteme membranare) obişnuite cu mult seturi de obiecte discrete, în baza cărora sunt introduse P sisteme hibride cu

multiseturi de obiecte discret-continue au fost introduse şi studiate în [4]. În mod similar cu comportarea P sistemelor hibride, acestea descriu procesele de calcul discret-continue care le vom folosi la modelarea şi evaluarea sistemelor de calcul mobil restructurabile. Un ghid complet pentru diferite extensii de P sisteme cu mult seturi de obiecte discrete poate fi referit la [2] şi pagina web [17].

Componentele de bază ale unui P sistem - sunt structuri de membrană ce constau din membrane ierarhic integrate în membrana exterioară. Fiecare membrană include o regiune ce conţine un multişor de obiecte discrete şi posibil alte membrane. Mai formal, o structură membranare este un arbore, ale cărui noduri sunt alte membrane, rădăcina este numită skin membrană, iar frunzele arborelui sunt numite membrane elementare. Informal, acestea pot reprezenta structuri membranale folosind diagrame Venn (Fig. 1).

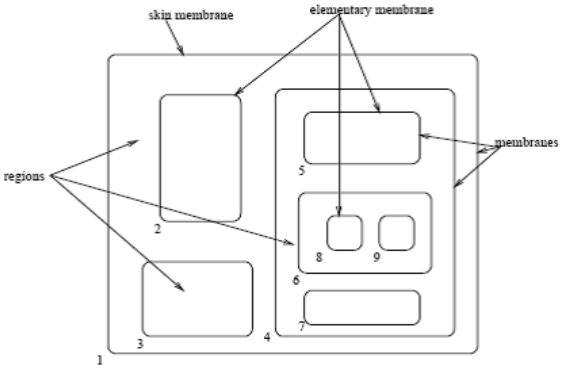

Fig. 1. Membrane structure

V. PETRI NETS EXPLORER

Petri Nets Explorer este o aplicaţie descktop pentru simularea animată a modelelor de reţele Petri reconfigurabile membranale. Aceasta are o interfaţă grafică intuitivă de creare, modelare şi simulare vizuală a reţelelor Petri membranale.

Fereastra de bază a acesteia este prezentată în Fig. 2.

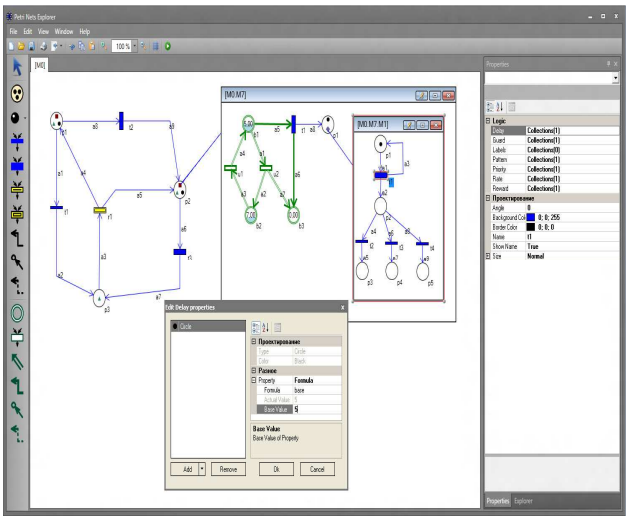

Fig. 2. Petri Nets Explorer

În partea stîngă se află meniul de instrumente numit „Tools”. Acest meniu conţine următoarele instrumente pentru crearea şi editare modelelor de reţele Petri:

Pointer – resetează instrumentul ales. Serveşte pentru selectarea elementelor de pe designer.

Discrete Place – creează locaţie discretă.

Token – adaugă token nou în locaţia discretă aleasă.

Immediate Transition – creează o tranziţie imediată.

Time Transition – creează o tranziţie temporizată.

Rewriting Immediate Transition – creează o tranziţie de rescriere imediată.

Rewriting Time Transition – creează o tranziţie de rescriere temporizată.

Normal Arc – creează un arc normal.

Inhibitor Arc – creează un arc inhibitor.

Test Arc – creează un arc de test.

Continous Place – creează o locaţie continuă.

Continous Transition – creează o tranziţie continuă.

Continous Arc – creează un arc continuu.

Flow Arc – creează un arc flow.

Inhibitor Arc – creează un arc inhibitor continuu.

Test Arc – creează un arc test continuu.

Fig. 3. Tools toolbar

Crearea ori editarea a modelului de reţea Petri înseamnă plasarea elementelor din toolbar şi interconectarea lor cu arce respective. Cînd se adaugă un element nou numele acestuia este generat automat după un format predefinit. Într-un model fiecare element are un nume unic după care

poate fi selectat in editorul de formule (Fig. 4).

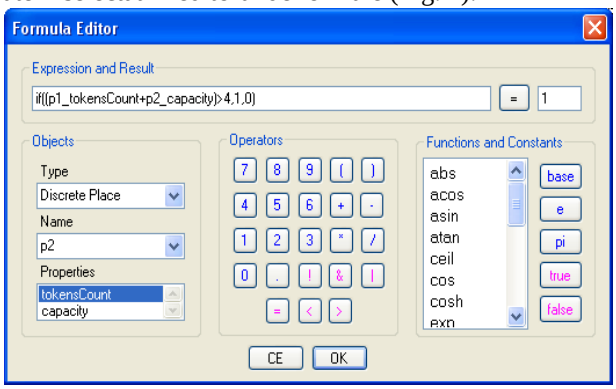

Fig. 4. Editor de formule

Fiecare element al modelului conține un set de proprietăți care pot fi editate de utilizator, astfel modificînd comportamentul modelului. Aceste proprietăți sunt accesibile în tabul Properties din dreapta.

Odată ce modelul a fost creat acesta poate fi simulat. Procesul de simulare este coordonat din fereastra Simulation (Fig. 4).

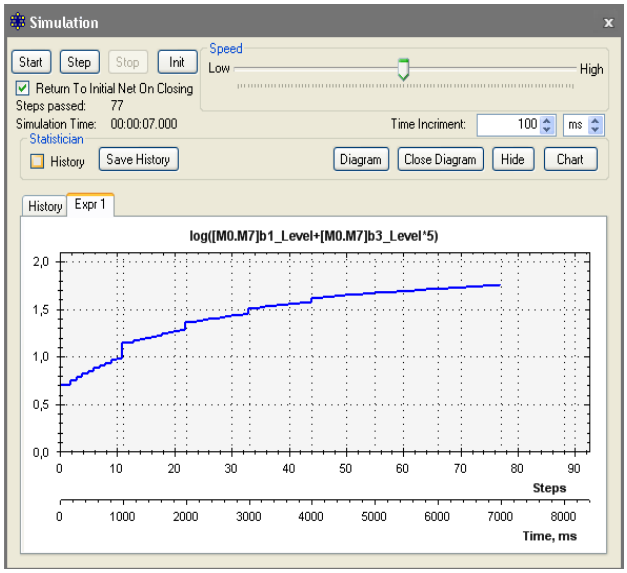

Fig. 5. Simulation Window

În timpul simulării modelului pot fi obținute date statistice referitor la comportamentul şi evoluția acestuia. Aceste date sunt prezentate în formă de diagrame. În baza cărora e mai uşor de înțeles procesul şi de a lua decizii sau formula concluzii la ipotezele setate în model.

VI. EXEMPLU DE MODEL

În Fig. 6 este prezentat un model de rețea Petri abstract al unui agent. Acest model consta din 3 locații discrete, 3 tranziții la fel discrete şi 6 arce normale. Este de menționat faptul că fiecare arc este descris nu de o singura valoare ci de o matrice de valori. Astfel configurarea ponderii arcelor devine mai flexibilă şi poate fi configurată să accepte valori marcaj dependente.

Pentru a asigura universalitatea sistemului în domeniul modelării agenților raționali este nevoie de a introduce calculul vectorial/matricial.

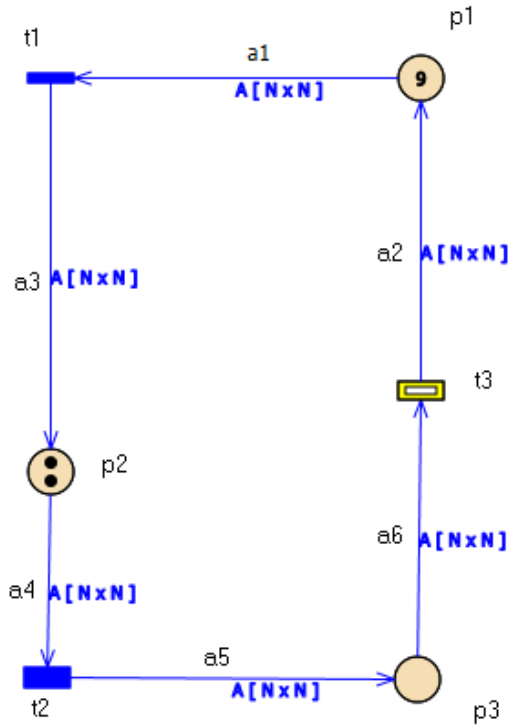

Fig.6. Model de rețea Petri cu parametri matriceali

Simbolic, matrice de valori sunt indicate ca fiind o etichetă cu dimensiunea **[N x N]**. Apăsînd dublu clic pe aceasta se deschide un dialog pentru editarea valorilor acestui parametru, cum este indicat în Fig. 7 ce urmează.

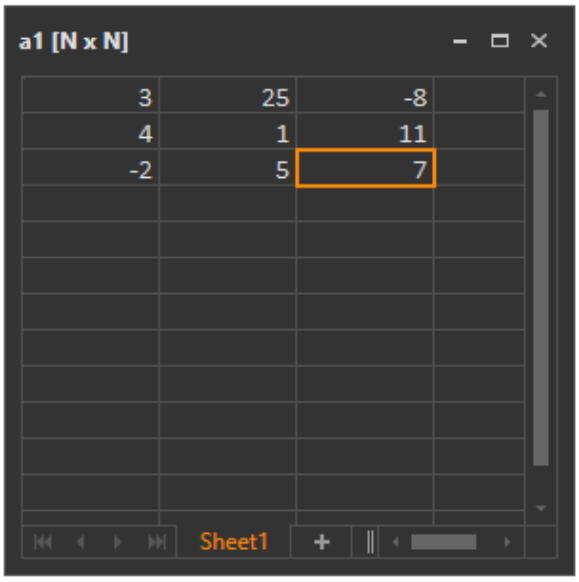

Fig.7. Model de rețea cu parametri matriciali

Acest mecanism permite utilizarea modelelor date pentru descrierea proceselor şi mai complicate, exemplu, comportamentul rețelelor neuronale.

Pentru a defini matriceal o rețea neuronală avem nevoie de o matrice bidimensională datorită faptului ca avem doar

două tipuri de entități noduri (neuroni) şi arce (axoni/dendride). Iar pentru a defini o rețea Petri matriceală avem nevoie de o matrice cel puțin tridimensională.

În acest scop în nucleul sistemului pentru simulare animată se introduce noțiunea de matrice şi aceasta trebuie să implementeze următoarea interfața.

```
public interface IMatrix<T>
    {
        IList<int> Size { get; set; }
        IList<T> Items { get; set; }
        T Value { get; }
        void SetValue(IEnumerable<int?>
indexes, IMatrix<T> value);
        IMatrix<T> GetValue(IEnumerable<int?>
indexes);

        IList<IMatrix<T>> Rows { get; }
        IList<IMatrix<T>> Columns { get; }
        IList<IMatrix<T>> Pages { get; }

        //Operations
        IMatrix<T> Inverse();
        IMatrix<T> Transpose();
        IMatrix<T> Add(IMatrix<T> other);
        IMatrix<T> Substract(IMatrix<T>
other);
        IMatrix<T> Multiply(IMatrix<T> other);

    }
```

Astfel, structura de date utilizată la implementarea interfeței grafice utilizator face posibilă păstrarea şi prelucrarea datelor multidimensionale (Fig. 8).

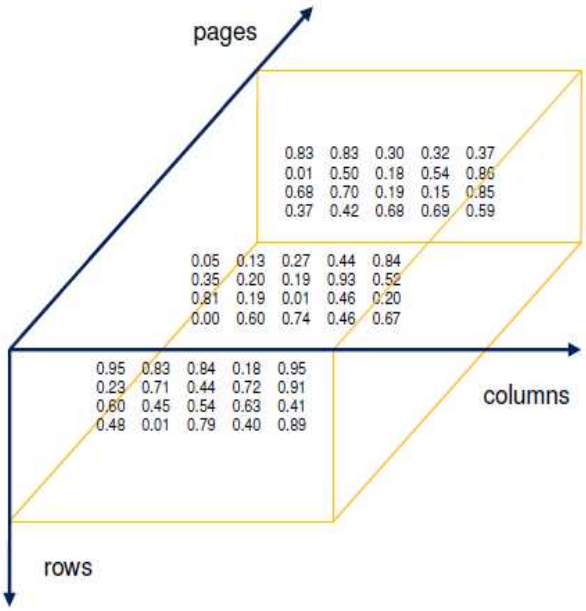

Fig.8. Matrice multidimensională

Una din problemele întâlnite este înmulțirea matricelor multidimensionale de diferite dimensiuni. Soluția acceptată spre implementare este MULTIPROD Framework (Multiple matrix multiplications, with array expansion enabled) publicat de Paolo de Leva în 2005.

VII. CONCLUZII

Datorită introducerii suportului pentru calcul vectorial modelele elaborate în Petri Nets Explorer pot servi pentru descrierea modelelor de agenți raționali în plină măsură, chiar dacă la baza implementării acestora stau alte formalisme ale științei calculatoarelor bazate pe elemente de inteligență artificială.

REFERENCES

[1] Guțuleac E. Evaluarea performanțelor proceselor de calcul prin rețele Petri generalizate stocastice. Meridian ingineresc, Nr. 1, 2008, Ed.: UTM, Chişinău, p. 27-32.

[2] Guțuleac E., Țurcanu I., Gutuleac Em. E. VMPN – software tool for performance modeling of dynamic modifiable structure systems with timed Membrane petri nets. Proceedings of the 9-th International Conference on DAS2008, 22-24 May 2008, Suceava, România, p. 149-155.

[3] Gaeta R., Manini M., and Sereno M. Stochastic Petri Nets Models for the Performance Analysis of TCP Connections Supporting Finite Data Transfer. Proceedings of Second International Wopkshop on Quality of Service in Multiservice IP Networks 2003, QoS-IP 2003, LNCS, No. 2601, Milano, Italy, Feb. 2003, Springer Verlag, p. 372-391.

[4] David R. Modelling Hybrid Systems using Continuous and Hybrid Petri Nets. Proceedings of International Conference PNPM'97, 1997, p.135-144.

[5] CALZAROSSA M. and MARIE R. Tools for Performance Evaluation. Performance Evaluation 33, 1998, p. 1-3.

[6] BULACEANU C.. Rețele locale de calculatoare. Editura Tehnică, Bucureşti, 1995, 415p.

[7] AJMONE-MARSAN M., BALBO G., and CONTE G. A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. ACM Trans. Computer Systems., vol. 2, no.2, may 1984, p. 93-122.

[8] MINER A.S., CIARDO G., and DONATELLI S. Using the exact state space of a Markov model to compute approximate stationary measures. Measurement and Modeling of Computer Systems, 2000, p.207–216.

[9] PĂUN Gh. Membrane Computing. An Introduction. Natural computing Series. ed. G. Rozenberg, Th. Back, A.E. Eiben .N. Kok, H.P. Spaink, Leiden Center for Natural Computing, Springer–Verlag, Berlin, 2002, p. 420.

[10] PĂUN GH., and ROZENBERG G. A Guide to Membrane Computing. Theoretical Computer Science, 287, 2002, p. 73-100.

[11] Petri nets world. http://www.informatik.uni-hamburg.de/TGI/PetriNets/.

[12] Petri nets world - Petri nets tools database. http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html

[13] Petri Nets Research Groups. http://www.informatik.uni-hamburg.de/TGI/PetriNets/research/

[14] P systems web page. http://psystems.disco.unimib.it.