SISTEM DE SIMULARE VIZUALĂ A PROCESELOR DE CALCUL ORIENTATE PE SERVICII PRIN REȚELE PETRI MEMBRANALE

Iurie Țurcanu, Alexei Cordunenu, Emilian Guțuleac
Universitatea Tehnică a Moldovei
alexei.corduneanu@yahoo.com, egutuleac@mail.utm.md

Abstract. *This paper presents a software environment VMPNS for visual simulation of reconfigurable service-oriented computing with membrane Petri nets. It offers an intuitive graphical user interface for designing various elements of nets as well as their efficient simulation, thus making it usable for research and academic activities.*

Cuvinte-cheie: *service-oriented systems, reconfigurable, membrane Petri nets, software environment, visual simulation*

I. Introducere

Actualmente, sistemele de calcul cu arhitecturi orientate pe servicii (SCOS) în timp real cunosc o dezvoltare rapidă, atât sub aspectul complexității şi/sau performanțelor, cât şi al ariei de răspândire [1, 2]. Acest tip de sisteme trebuie să aibă o flexibilitate, disponibilitate şi siguranță în funcționare deosebită. Astfel, un sistem SCOS poate fi considerat şi implementat drept fiind o colecție de configurații, unde fiecare din acestea este o rețea de componente ce comunică între ele. Diferite configurații pot fi folosite pentru procesarea a diferitor servicii sau condiții de operare ale aplicațiilor. Ca urmare, facilitățile trecerii în timp real de la o configurație către alta pe parcursul rulării, duc la creşterea flexibilității sistemului [3, 10]. De asemenea, aceste facilități ale configurațiilor ce prestează noi servicii şi ale eliminării configurațiilor serviciilor care nu mai sunt necesare, reduc perioadele de remediere în caz de defectare, crescând astfel disponibilitatea sistemului. Înlocuirea configurației de executare atunci când se manifestă un comportament eronat cu o configurație validă duce şi la creşterea s*iguranței în funcționare* a sistemului şi deci la creşterea fiabilității şi previzibilității acestuia, asigurarea căruia în timpul reconfigurării dinamice este dificilă din cauza interacțiunii între serviciile de aplicație, care sistemul le oferă utilizatorului.

Unul dintre cele mai răspândite formalisme moderne, folosite pentru modelarea şi analiza sistemelor paralele/distribuite cu evenimente discrete, sunt rețelele Petri de diferite extensii [8, 12, 13, 14]. Totodată, apare necesitatea de a dezvolta aceste formalisme pentru a descrie mai adecvat, mai flexibil şi mai comod sisteme SCOS cu structuri complexe dinamic restructurabile. De aceea, actualmente sunt efectuate intens cercetări pentru a extinde formalismul rețelelor Petri în baza conceptului orientat obiect şi servicii pentru a obține modele, care explicit descriu (auto)reconfigurabilitatea şi restructurarea ierarhică a sistemului.

Problemele menționate au fost abordate în lucrările recente [6, 7, 9], unde în baza conceptului de P sisteme, adică a sistemelor membranale [11], au fost introduse şi studiate rețele Petri hibride temporizate (RPHT) reconfigurabile (RPHR) şi rețele Petri membranale (RPM) [4, 5], care permit de a descrie structurarea modulară, organizarea ierarhică, reconfigurabilitatea proceselor de calcul şi mobilitatea lor.

Pentru a avea posibilitatea de a efectua în mod automat verificarea funcțională, simularea vizuală şi evaluarea indicatorilor de performanță specificate în baza modelelor acestor tip de rețele Petri a fost elaborat un produs software instrumental, numit simulator VMPNS al proceselor de calcul orientate pe servicii reconfigurabile. În continuare, succint sunt prezentate unele elemente ale

rețelelor Petri hibride reconfigurabile (RPHR), rețelelor Petri hibride membranale şi aspecte de elaborare, realizare şi utilizare ale VMPNS.

II. Rețele Petri hibride reconfigurabile

În rețele RPHR *dinamic descriptiv-restructurabile* [6, 7] este introdusă o mulțime de reguli de rescriere $R = \{r_1, ..., r_k\}$ a rețelei curente care poate modifica atât marcajul, cât şi structura ei la ocurența unor evenimente specificate. O regulă de rescriere $r_j \in R$ este o generalizare a noțiunii de tranziție discretă $t_j \in T_d$, folosită în sens clasic. Condiția de validare de către marcajul curent M al unor reguli $r_j \in R$ sunt similare cu cele ale tranzițiilor $t_j \in T_d$.

În fig. 1 sunt prezentate primitivele unei rețele RPHR.

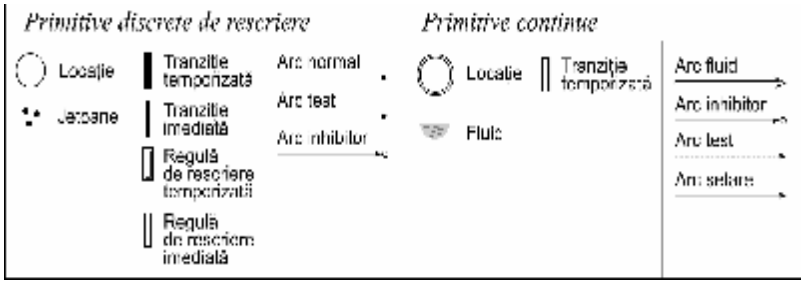

Fig.1. Primitivele unei rețele RPHR

Validarea unei reguli r în RN este efectuată cum şi pentru partea discretă a RPHT. La declanşarea lui r, în cazul în care funcția de gardă $g_r(M)$ asociată cu r are valoarea "*false*", r va schimba numai marcajul curent M al RN într-un marcaj nou M'. Însă dacă $g_r(M)$ are valoarea "*true*", r va reconfigura rețeaua RN în altă rețea RN' după regula de rescriere $r : RN_L > RN_W$. Modificarea retelei RN are loc în două etape. La prima etapă are loc eliminarea elementelor specificate de subrețeaua RN_L ale rețelei curente, obținând o nouă rețea $RNr = RN \setminus RN_L$. Etapa a doua constă din adaugarea subrețelei RN_W în rețeaua RNr curentă cu elemente noi ale rețelei, obținând astfel $RN' = RNr \cup RN_W$. Dacă în rețeaua curentă sunt elemente cu acelaş nume ca la elementele ce urmează a fi adaugate, ele se contopesc. Astfel, ca rezultat se va obține o rețea nouă RN', care va funcționa după noi reguli.

III. Rețele Petri hibride membranale

Sistemele SCOS au o structură modulară în care însăşi resursele reprezintă un subsistem cu un comportament propriu. Această proprietate poate fi descrisă de către RPM. Aceasta reduce complexitatea modelului şi măreşte lizibilitatea lui.

Rețelele RPM sunt o generalizare a rețelelor RPHR, folosind paradigma de P sisteme hibride ΓP [?], membranele căruia sunt structurate prin subrețele *RDMR*, expresia descriptivă *DE* a cărora este determinată [?]. Un ghid complet pentru diferite extensii de P sisteme cu multiseturi de obiecte discrete poate fi referit la pagina web: *http://psystems.disco.unimib.it/*.

Pentru a reprezenta compartimentarea RPHR într-o membrană din ΓP, sunt folosite tranziții $t_{h,i}$ *localizate* în membrana *h*, notată $[_h \quad]_h$, cu execuții locale maximal concurente ale tranzițiilor co-locate în aceeaşi membrană şi, de asemenea, locații $p_{h,i}$ *localizate* în aceeaşi membrană [105].

Maparea ΓP într-o rețea RPM este constituită din doi paşi separați. Mai întâi, pentru fiecare membrană $[_h \quad]_h$, este asociată: (i) fiecărui obiect $w_{h,i} \in w_h$ în membrana h o locație $[_h \; y_{h,i}^0 p_{h,i}]_h$ cu

marcajul inițial $y^0_{h,i} \in \{ m^0_h(p_i), x^0_h(b_i)\}$ al locației $p_{h,i} \in P^h$ şi fiecărei reguli de tip obiect continue $r^c_{h,j}$ o tranziție continuă $[_h\ u_{h,j}\ ]_h$, $u_{h,j} \in T^h_C$ co-locată în membrana *h*; (ii) fiecărei reguli de tip obiect discret $r^{do}_{h,j}$ sau fiecărei reguli de rescriere $r^{dm}_{h,j}$ asociem un eveniment discret $[_h\ e_{h,j}\ ]_h$, $e_{h,j} \in E^h$ ce acționează în această membrană. În al doilea plan, pentru fiecare membrană $[_h\ \ ]_h$ definim o expresie descriptivă DE^0_h [4, 5] a subrețelei RN^0_h ce corespunde configurației inițiale a lui ΓP, ca fiind o subrețea membranală $[_h\ DE^0_h\ ]_h$.

O rețea RPM de gradul $n \geq 0$, este o construcție structurată $RPM = \vee^{n-1}_{h=0}([_h\ DE^0_h\ ]_h)$ de subrețele DE^0_h cu setul de reguli de rescriere ale obiectelor şi membranelor $MR = \{ mr_i,\ i = \overline{1,n}\}$, de exemplu:

- mr_0: regula *Change* de *schimbare* a conținutului membranei $[_{h'}\ DE_{h'}\ ]_{h'}$ care, la procesarea ei, schimbă structura curentă şi multisetul de obiecte în membrana h', redată de expresia descriptivă $DE_{h'}$ şi marcajele M_h, într-o nouă structură $DE'_{h'}$ cu noi marcaje $M'_{h'}$;
- mr_1: regula *Dissolve* care *dizolvă* membrana h'. Ca rezultat, obiectele $M_{h'}$ şi sub-membranele care aparțin membranei h' vor aparține membranei pârinte *h*, skin membrana nu poate fi dizolvată: mr_1: $[_h DE_h\ [_{h'}\ DE_{h'}\ ]_{h'}\ ]_h > [_h\ \ DE'_h\ ]_h$ cu $M'_h = M_h + M_{h'}$;
- mr_2: regula *Create*, care creează o nouă membrană h' cu $DE''_{h'}$ şi $M''_{h'}$, restul conținutului ramâine neschimbat în membrana părinte *h*;
- mr_3: regula *Divide* de *divizare* la rescriere a membranei *h*. Ca rezultat, obiectele şi sub-membranele ei sunt reproduse şi adăugate respectiv în membrana h' şi h'';
- mr_4: regula *Merge*. Obiectele membranei h' si h'' sunt adăugate unei noi membrane *h*;
- mr_6: *regula Separate*, fiind opusul regulii *Merge*, la rescriere va fi efectuată *separarea* membranei *h* în două membrane h' şi h'';
- mr_6: regula *Move* conform căreia, la rescriere, membrana h'' va fi mutată în afara sau în interiorul membranei h' cu marcajele lor, respective.

Conceptorul poate să definească şi alte tipuri de reguli *MR*, dacă apare necessitatea de a exprima alte proprietăți de comportare.

Astfel, folosind rețelele RPM putem obține o descriere compactă şi flexibilă, modulară şi ierarhică a specificațiilor şi modelelor discret-continue de rețele RPHR pentru verificarea, simularea şi evaluarea performanțelor sistemelor de calcul mobil reconfigurabile.

IV. Descrierea sistemului VMPNS

Sistemul software instrumental VMPNS este o aplicație desktop şi dispune de o interfață grafică User-friendly. Acest sistem a fost elaborat pe platforma Microsoft .NET, ceea ce permite realizarea într-un timp relativ scurt a aplicațiilor complexe.

În continuare, sunt descrise formele de bază ale VMPNS, precum şi modul de creare a modelelor de rețele Petri membranale, redactarea, setarea parametrilor elementelor acestora, lansarea simulării şi colectarea datelor statistice.

Interfața Grafică a Utilizatorului (GUI) este prezentată în fig. 2**.** Pentru a lansa o rețea putem să o creăm sau să deschidem o rețea existentă din colecția de rețele de lucru afişată în fereastra *Nets*. În rezultatul lansării unei rețele existente se va deschide o fereastră în care se prezintă rețeaua, cum este prezentat în fig. 2. Observăm că fereastra GUI conține 3 meniuri de bază: meniul principal, meniul comenzilor şi meniul instrumentelor de creare/editare a rețelei. Meniul principal este construit după un şablon clasic. Este de menționat că la deplasarea cursorului asupra butonului apare descrierea acestuia. La stânga GUI, de jos în sus, în meniul dat sunt prezentate instrumentele disponibile (fig.3) pentru crearea modelelor de rețele: *Pointer* – dezactivarea tuturor instrumentelor;

Discrete Place – crearea locației discrete; *Token* – adăugarea jetonului într-o locație discretă; *Immediate Transition* – crearea tranziției imediate; *Time Transition* – crearea tranziției temporizate; *Rewriting Immediate Transition* – crearea regulei de reconfigurare imediate; *Rewriting Time Transition* – crearea regulei de reconfigurare temporizate; *Normal Arc* – crearea arcului discret; *Inhibitor Arc* – crearea arcului inhibitor discret; *Test Arc* – crearea arcului test discret; *Continous Place* – crearea locației continue; *Continous Transition* – crearea tranziției continue; *Continous Arc* – crearea arcului continuu; *Flow Arc* – crearea arcului fluid; *Inhibitor Arc* – crearea arcului inhibitor continuu; *Test Arc* – crearea arcului test continuu.

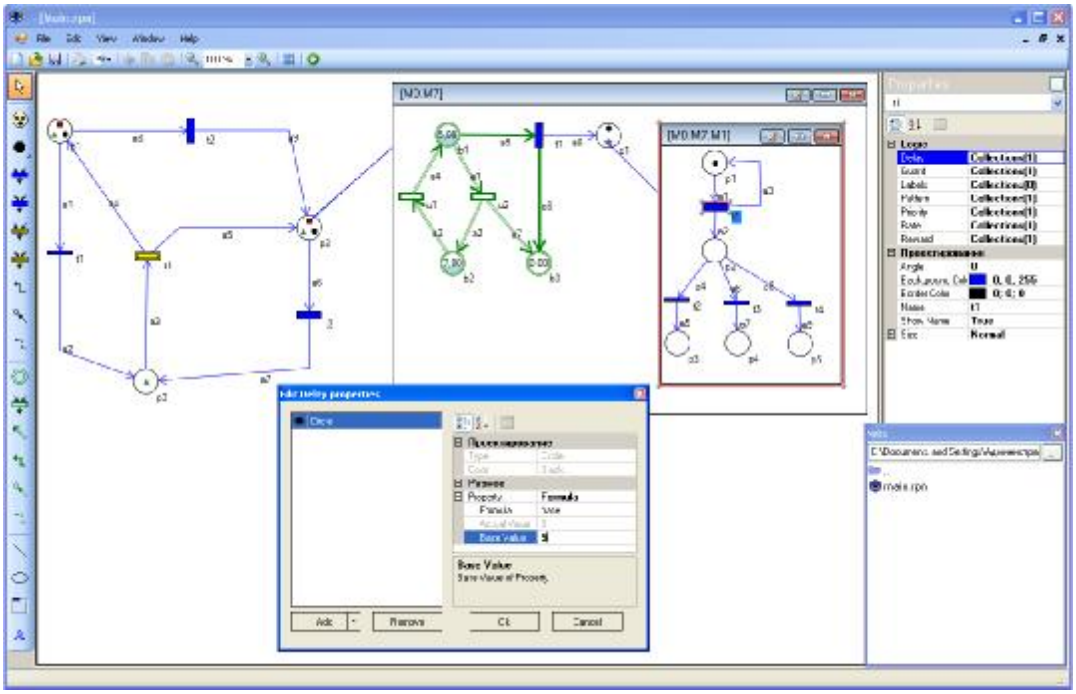

Fig.2. Interfața Grafică Utilizator a sistemului VMPNS

La deplasarea cursorului asupra instrumentului respectiv apare denumirea acestuia.

Fig.3. Meniul de instrumente.

Pentru comoditatea utilizatorului este realizat un meniu de context care se activează la apăsarea *click drept* pe forma de redactare a rețelei. Meniul dat conține la fel unele din comenzi de bază ale aplicației, dar și proprietățile specifice ale unor obiecte.

Primele trei operații efectuează aceleași operații ca și în meniul principal. Comanda *Delete* șterge obiectul/obiectele selectate. Comanda *Size* permite setarea dimensiunilor potrivite pentru obiectele selectate, aceasta fiind proprie doar locațiilor și tranzițiilor. Comanda *Rotate* este proprie doar tranzițiilor și permite rotirea acestora în jurul centrului său cu un unghi de 45°.

Formele secundare ale aplicației. La formele secundare ale aplicației se referă în primul rând forma de editare a proprietăților obiectelor din rețea (*Properties*) și forma de vizualizare a rețelelor de lucru. Aceste ferestre se vizualizează la *activarea* aplicației. La fel există și un set de forme care sunt pasive la lansarea aplicației. În continuare vom analiza ambele tipuri de ferestre secundare.

Editorul proprietăților obiectelor. În fig. 4 este prezentată forma de editare a proprietăților obiectelor rețelei. Implicit ea se poziționează în colțul dreapta sus al formei de bază, dar poate fi repoziționată, sau închisă. Această formă este compusă din următoarele 4 componente: lista obiectelor existente în rețea; butoanele de gestiune ale modului de vizualizare a proprietăților

obiectului selectat (cu grupare pe categorii sau fără); câmpul de descriere a proprietății selectate. La selectarea obiectului din editorul rețelei, el automat se selectează în fereastra *Properties*. Toate proprietățile se împart în două clase: *Logic* şi *Design*.

Editorul de formule. Pentru a edita rețele cu automodificare a fost realizat un editor de formule care are posibilitatea de a reda unele proprietăți cantitative ale atributelor rețelei. Acesta conține setul întreg de operații şi funcții matematice uzuale, precum şi operatorul condițional *if*.

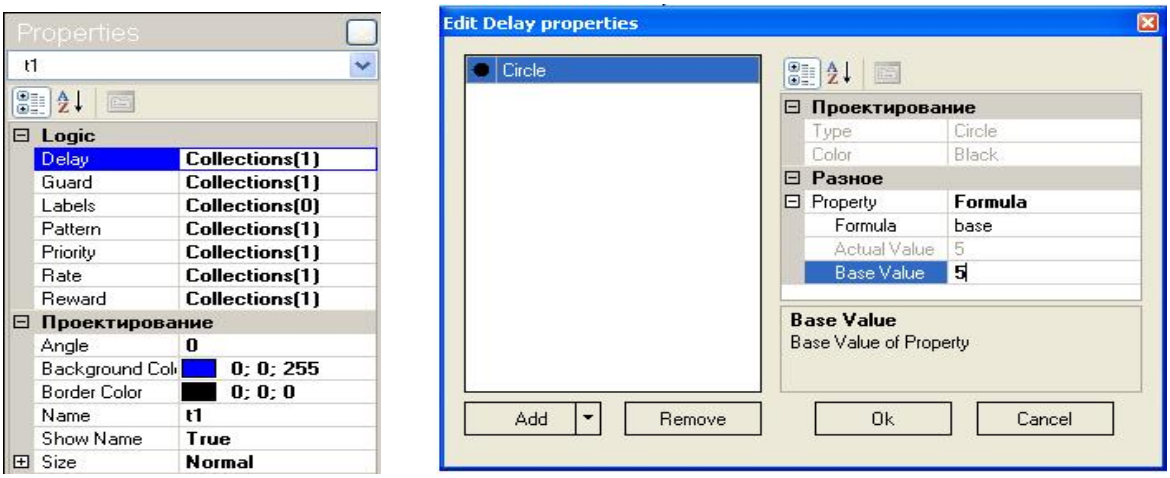

Fig.4. Redactarea proprietăților elementelor rețelei

Editorul de formule. Pentru a edita rețele cu automodificare a fost realizat un editor de formule care are posibilitatea de a reda unele proprietăți cantitative ale atributelor rețelei. Acesta conține setul întreg de operații şi funcții matematice uzuale, precum şi operatorul condițional *if*.

Forma de dirijare a simulării. O altă fereastră secundară a sistemului este forma „*Simulation*", prezentată în fig. 5. Ea se lansează la alegerea butonului *„Run"* al meniului principal.

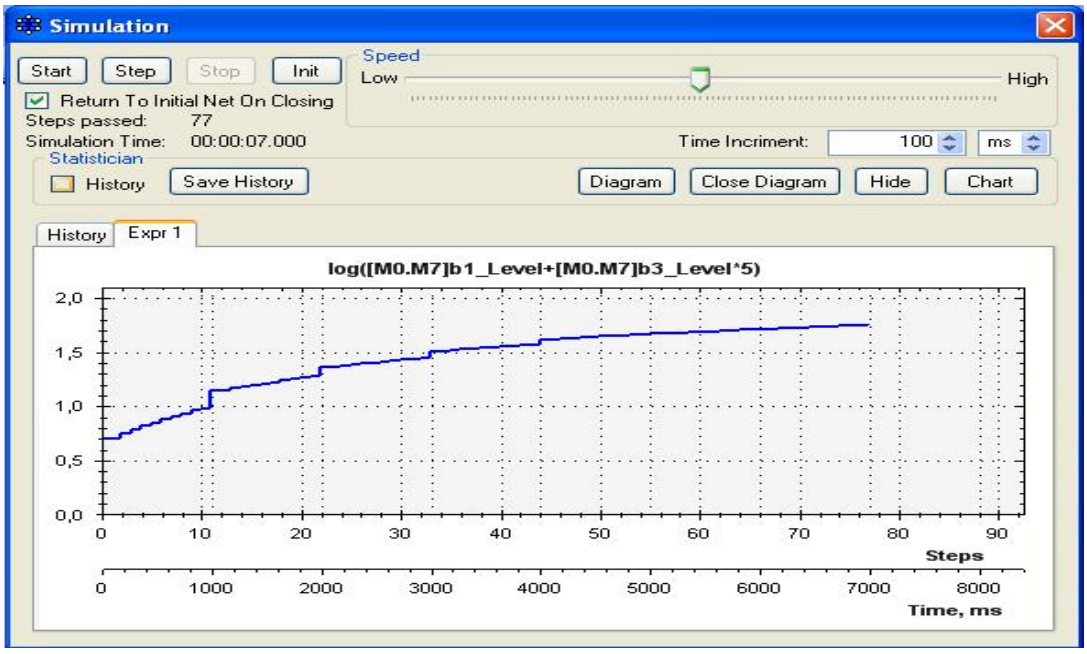

Fig.5. Diagramă de simulare

În partea superioară a formei se află următorul grup de butoane: *Start* – startează procesul de simulare. Dacă o simulare anterioară a fost oprită butonul Start o va prelungi; *Step* – efectuează un singur pas de simulare; *Stop* – opreşte procesul de simulare; *Init* – reinițializează rețeaua dacă ea a efectuat cel puțin un pas de simulare.

În partea sus - dreapta se află regulatorul vitezei procesului de simulare – *Speed*. Imediat sub acest regulator se află regiunea de setare a quantei de timp / pás care arată câte unități de timp decurg la un pas de simulare. În partea de jos a formei este situat grupul de butoane *Statistic*. Funcția grupului constă în colectarea datelor statistice. Primul din acestea este indicatorul „*History*", care permite de a păstra istoria simulării dacă este nevoie. Istoria simulării este prezentată sub forma unui tabel în care sunt stocate aşa date ca: pasul simulării; timpul simulării; marcajul curent; tranzițiile validate şi cele declanşate.

Crearea unei rețele constă în plasarea unui set de elemente pe planşeta formei respective, interconectarea lor şi setarea proprietăților lor. La adăugarea unui element nou, lui i se atribuie un nume după şablonul specificat. Fiecare din elementele rețelei mai conține şi un set de proprietăți care pot fi setate de utilizator

La simulare rețelei tranzițiile validate vor fi colorate în roşu, iar cele declanşate în verde. Procesul de declanşare a unei tranziții validate este însoțit de deplasarea jetoanelor pe arce discrete sau scurgerea fluidului prin arce continue.

V. Concluzii

Rețelele Petri hibride membranale sunt un instrument adecvat la modelarea funcționării proceselor discret-continue ale sistemelor de calcul orientate pe sevicii reconfigurabile.

În lucrare sunt prezentate unele aspecte de elaborare şi utilizare a unui sistem software instrumental, VMPNS., realizat pentru modelarea, verificarea funcțională, simularea animată şi evaluarea performanțelor proceselor discret-continue de calcul orientate pe servicii reconfigurabile, descrise prin rețelele Petri membranale.

VI. Referințe

1. Bell M., Service-Oriented Modeling (SOA): Service Analysis, Design, and Architecture. –Wiley, 2008. –320 p.
2. Compton K., Hauck S. Reconfigurable Computing: a Survey of Systems and Software. ACM Computing Surveys (CSUR), vol. 34, no. 2, 1998, pp. 171-210.
3. Guțuleac E., Mocanu M. L., Țurcanu Iu. Membrane Differential Petri Nets for Performance Modelling of Hybrid P-Systems. Control Engineering and Applied Informatics, vol. 9, no. 3-4, Ed.: SRAIT, Bucureşti, România, 2007, pp. 12-21.
4. Guțuleac E. Dynamic rewriting generalized differential Petri nets for discrete-continuous modeling of computer systems. Meridian ingineresc, Nr. 3, Ed.: UTM, Chişinău, 2006, p.27-32.
5. Guțuleac E. Evaluarea performanțelor sistemelor de calcul prin rețele Petri stochastice. –Tehnica-Info, Chişinău, 2004, – 276p.
6. Llorens M., Oliver J. Structural and Dynamic Changes in Concurrent Systems: Reconfigurable Nets, IEEE Transactions on Computers, vol. 53, no. 9, 2004, pp. 1147-1158.\
7. Papazoglou M.P., Van den Heuvel W-J. Service-Oriented Architectures. Approaches, Technologies and Research Issues, VLDB J., vol. 16, no. 3, 2007, pp. 389-415.
8. Păun Gh., and Rozenberg G. A Guide to Membrane Computing. Theoretical Computer Science, 287, 2002, pp. 73-100.
9. Petri nets world. *http://www.informatik.uni-hamburg.de/TGI/PetriNets/*
10. Petri nets world - Petri nets tools database. *http://www*.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html.
11. Tsai W. T., Cao Z., Wei X., Paul R., Huang Q., Sun, X. Modeling and simulation in service-oriented software development. Simulation Journal, 83(1), 2007, pp. 7-32.