Compunerea în Mediul PIPE a Modelelor de Rețele Petri prin Expresii Descriptive

Emilian GUȚULEAC
dr. hab. şt. tehnice, prof. univ.
Universitatea Tehnică a Moldovei
egutuleac@mail.utm.md

Anatolie CRUCEAN
inginer
Universitatea Tehnică a Moldovei
cruceananatolie@gmail.com

Andrei STAVER
inginer
staver.andrei@yahoo.com

***Abstract* —În lucrare sunt considerate unele aspecte de elaborare şi implementare în limbajul Java a unui subsistem program pentru compunerea modelelor de rețele Petri stocastice generalizate (GSPN) prin expresii descriptive care permit de a formaliza etapa de trecere logică de la o descriere informală a arhitecturii şi a specificațiilor comportamentale ale sistemului analizat la maparea lor în modele GSPN. Acest subsistem este integrat în mediul PIPE v4.30 (Independent Petri Net Editor Open-Source), care pe lângă crearea redactarea, simularea, analiza comportamentală şi facilităților de animație a modelelor GSPN, el prevede şi un mecanism de integrare run-time a noi funcționalități printr-un modul pluggable de analiză.**

***Index Terms* — compunere, modelare, reȚele Petri stocastice, mediu de simulare.**

I. INTRODUCERE

Analiza performanțelor constituie una din componentele importante ale activităților de elaborare, realizare şi întreținere ale sistemelor de calcul. La nivelul activităților de proiectare şi realizare, interesul estimării performanțelor viitoare ale noilor sisteme de calcul tinde să crească în condițiile sporirii complexității sistemelor de realizat şi implicit a *riscului* de a obține produse insuficient adaptate destinației şi cerințelor de performanțe propuse. În mod asemănător, alegerea unui sistem de calcul adecvat cerințelor proprii unei clase de aplicații presupune investigații preliminare privind comportarea sistemelor disponibile în contextul viitor de utilizare [1, 2].

Unul dintre cele mai răspândite formalisme moderne, folosite pentru modelarea şi analiza sistemelor paralele/distribuite cu evenimente discrete, sunt rețelele Petri (RP) de diferite extensii. Ca unelte grafice şi matematice, RP asigură un mediu uniform pentru modelare, analiză formală şi proiectare a sistemelor cu evenimente discrete (SED). Principalul avantaj al folosirii RP îl constituie faptul că același model poate fi folosit atât pentru analiza proprietăților comportamentale şi evaluarea performanțelor, cât şi pentru construcția sistematică a simulatoarelor şi controlerelor cu evenimente discrete. Ca unealtă grafică, RP asigură şi un puternic mediu de comunicare între utilizator şi client.

De asemenea, cerințele complexe din caietele de sarcini pot fi reprezentate grafic, folosind RP în locul unor descrieri textuale ambigue sau al unor notații matematice dificil de înțeles de către client. Acest aspect, combinat cu existența unor medii software instrumentale, care permit simularea grafică interactivă a modelelor de RP, asigură inginerilor de dezvoltare o unealtă puternică ce să îi asiste în procesul de proiectare al SED.

Totuşi, motivul pentru care RP sunt folosite mai mult în mediul academic şi în instituțiile de cercetare constă în dificultatea construcției modelelor. Aceasta necesită o mare experiență, mai ales pentru sistemele complexe şi foarte mari. Nu există metodologie disponibilă, pentru a automatiza în vreun fel procedeul de construcție a modelelor de RP, ci acestea sunt concepute într-o manieră ad-hoc. Astfel, folosirea pe scară largă a RP, mai ales în industrie, va trebui să fie susținută de metode şi unelte suport care să permită o construcție automată sau semiautomată a modelelor pe baza specificațiilor de dezvoltare. Pentru a facilita punerea în practică a acestui deziderat, în [3, 4] a fost introdusă noțiunea de *dexel* cu atributele unei RP primitive, definite un set de operații compoziționale şi expresii descriptive, care permit de a formaliza etapa de trecere logică de la descrierea informală a arhitecturii, specificațiilor comportamentale ale sistemului studiat, la determinarea unor expresii descriptive şi la maparea lor direct în RP.

II. EXPRESII DESCRIPTIVE ALE GSPN

Cu toate că, actualmente, rețelele Petri de diferite extensii sunt recunoscute ca un puternic şi intuitiv formalism de modelare a sistemelor cu evenimente discrete concurente şi paralele, lipsa *construcțiilor compoziționale*, integrate în astfel de modele, face ca utilizarea lor să fie dificilă şi, deseori, nepotrivită la modelarea diverselor sisteme reale. Compunerea acestor tip de modele prin rețele Petri plate, devine mai degrabă o artă decât o rutină.

Pentru a reda proprietați compoziționale modelelor de rețele GSPN, în mod similar cu noțiunea de *pixel*, in [3] este introdusă *noțiunea* de *dexel* (***d****escriptive* ***ex****pression* ***el****ement*) şi un set de operații compoziționale cu atribute respective care permit de a construi expresii descriptive, care sunt mapate direct in modelul specificat de conceptor.

În continuare, a facilita expunerea lucrării date, prezentăm succint unele operații compoziționale. Mai detaliat cititorul poate consulta lucrarea [3, 4].

Elementul de expresie descriptivă primitivă *bDE* , al *GeN* primitive *b*N subiacente GSPN, numit *dexel*, este :

$$bDE = {}^{\Pi_j}_{g_j}|^{\alpha_j}_{t_j}\, m_{0i}{}^{k_i}\, y_i^{\beta_i}\, [W_i^+, W_i^-]\, {}^{\Pi_k}_{g_k}|^{\alpha_k}_{t_k},$$

unde $y \in \{p, \overline{p}, \widetilde{p}\}$ este simbolul-locație ce determină respectiv tipul de arc ({*normal*, *inhibitor*, *test*}) cu ponderea $W_i^- \in \{\Pr e(t_k, p_i),\ Inh(t_k, p_i),\ Test(t_k, p_i)\}$ incident

inainte la tranziția $|_{t_k}$, iar $W_i^+ \in \{Post(t_j, p_i)\}$ este ponderea arcului normal ce iese din tranziția $|_{t_j}$ și intră în locația p_i. Atributele p_i respectiv sunt: m_{0i} -marcajul inițial; k_i - capacitățile locației; β_i -eticheta locației ce redă tipul de condiții. Atributele tranzițiilor t_j și t_k respectiv sunt: g_j și g_k - funcțiia de gardă; Π_j și Π_k - funcția de prioritate; α_j și α_k - eticheta tranziției ce redă tipul de acțiune sau activitate. Unele atribute pot fi omise, de exemplu: $m_{0i} = 0$; $K_{p_i}^{\min} = 0$, iar $K_{p_i}^{\max}$ este considerată nelimitată; $\beta_i = \{\varnothing\}$, $\alpha_j = \alpha_k = \{\varnothing\}$; $g_j = g_k =$"*true*"; $\Pi_j = \Pi_k = 0$. În cazul în care $W_i^- = W_i^+ = 1$ paranteza pătrată se va omite.

Maparea unor derivative ale *bDE* în rețele *GeN* primitive *b*N este prezentată în Fig. 1.

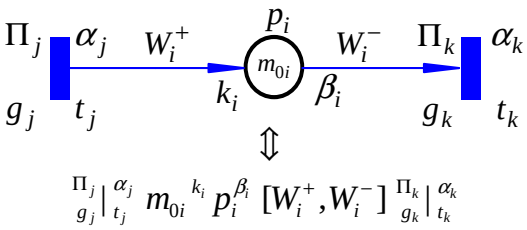

Fig. 1. Maparea unor derivative ale *bDE* în rețele *GeN* primitive *b*N.

Cu ajutorul diferitor derivative ale *bDE* și operații *compoziționale unare* și/sau *binare*, folosind un raționament adecvat ce redă interacțiunea condițiilor și evenimentelor sistemului specificat, putem compune expresii descriptive ale modelelor *GeN* (sub)sistemelor considerate.

O expresie descriptivă (*DE*) a unei rețele N tip *GeN* este:

$$DE ::= bDE \mid DE_i * DE_j \mid \circ DE,$$

unde $*$ reprezintă operatorul unei *operații binare*, iar $\circ$ reprezintă operatorul unei operații unare. Implicit, la aplicarea acestora locațiile și tranzițiile ce au același nume se vor contopi respectiv.

Într-o *DE*, orice simbol-locație sau simbol-tranziție poate fi folosit în orice ordine de mai multe ori. Astfel, se va subînțelege că în rețele *GeN* respective, redate de expresia *DE*, aceleași locații (tranziții), cu același simbol vor fi *contopite* într-un singur *simbol-locație* (*simbol-tranziție*). Ca rezultat al aplicării acestor operații compoziționale, obținem o nouă clasă de subrețele N_i, interconectarea cărora, conform *DE*, va determina rețeaua N rezultantă, redată de această expresie.

Redăm în continuare unele operații compoziționale:

- *Operația Inhibiție*, redată de *operatorul* "$\overline{p}_i$", este o operație unară. Ea descrie faptul că la ocurența pre-condițiiei p_i, nu mai poate avea loc ocurența evenimentului specificat. Acestei operații îi corespunde dexel-ul $DE1 = m_{0i}{}^{k_i}\,\overline{p}_i^{\beta_i}\,[W_i]\,{}^{\Pi_k}_{g_k}|^{\alpha_k}_{t_k}$ ce redă o rețea primitivă, constituită din locația p_i și arcul inhibitor ce duce din această locație în tranziția t_k, cu atributele respective, legată de declanșarea evenimentului specificat.

- *Operația Test* cu *operatorul* "$\tilde{p}_i$", descrie o buclă a rețelei impure, redată de $DE2 = m_{0i}\tilde{p}_i\,[W_i]\,|^{\alpha_j}_{t_j}$, ea reprezintă operația unară cu test arc.

- *Operația Sincronizare*, redată de *operatorul* "$\bullet$" sau "$\wedge$", este o operație binară *comutativă*, *asociativă* și *reflexivă* ce descrie *sincronizarea* pre-condițiilor legate cu $p_i \in {}^\bullet t_j$, ale unui eveniment t_j, apariția căruia va avea loc numai atunci, când concomitent aceste pre-condiții sunt verificate, descrise de *DE*3:

$$DE3 = (m_{01}{}^{K_{p_1}} y_1\,[W_1] \bullet \ldots \bullet m_{0n}{}^{K_{p_n}} y_n\,[W_n])\,|^{\alpha_j}_{t_j}$$
$$= (\bigwedge_{i=1}^{n} m_{0i}{}^{K_{p_i}} y_i\,[W_i])\,|^{\alpha_j}_{t_j}.$$

- *Operația Secvențialitate*, redată de *operatorul* **"|"**, este o operație binară ce determină logica "cauză-consecință" a relației dintre două stări locale p_i (pre-condiție) și p_k (post-condiție), determinată de acțiunea t_j. Această o*perația*, exprimată de expresia *DE4*, este *asociativă*, *reflexivă* și *tranzitivă*, însă *necomutativă*: $DE4 = m_{0i}\,p_i\,[W_i]\,|^{\alpha_j}_{t_j} m_{0k}\,p_k\,[W_k] \neq m_{0k}\,p_k\,[W_k]\,|^{\alpha_j}_{t_j} m_{0i}\,p_i\,[W_i]$.

Modelarea operației **"*iterație*"** poate fi redată prin contopirea locațiilor de la începutul expresiei cu cea de la sfârșitul ei care au același nume (operația **"*închidere*"**).

- *Operația AND-Split sau Fork*, redată de *operatorul* "$\Diamond$" sau "**;**" , descrie faptul că la apariția unui eveniment specificat t_j se vor produce concomitent două sau mai multe post-condiții. Aceasta operație binară, fiind *comutativă*, *asociativă* și *reflexivă*, este redată de următoarea expresia *DE5*:

$$DE5 = |^{\alpha_j}_{t_j}(\, m_{01}p_1\,[W_1] \Diamond \ldots \Diamond m_{0n}p_n\,[W_n]\,)$$
$$= |^{\alpha_j}_{t_j}(\,(\Diamond)_{k=1}^{n} m_{0k}\,p_k\,[W_k]\,)\,.$$

- *Operația Paralelism competitiv*, redată de *operatorul* "$\vee$" sau "+", descrie *relațiile logice de paralelism competitiv* ale condițiilor și evenimentelor între două sau mai multe procese concurente. Ea este aplicată pentru a efectua compunerea unor submodele de subrețele GSPN, ce descriu funcționarea subsistemelor respective, într-un model rezultant al sistemului considerat. Fie două subrețele N_A și N_B sunt redate de expresiile respective $DE_A = A$ și $DE_B = B$, atunci la compunerea lor prin aplicarea operatorului "$\vee$", relativ la aceste două expresii descriptive, obținem o rețea rezultantă N_R redată de $DE_R = R = A \vee B$ în care locațiile și tranzițiile ce au același nume, respectiv, vor fi contopite. Nodurile contopite vor păstra atributele și incidența arcelor din fiecare subrețea. Această operație este *comutativă*, *asociativă* și *reflexivă*.

Având o descriere informală, aceasta implică faptul că la elaborarea modelului de rețea *GeN* este indispensabil de a cunoaște:

– structura, componentele sistemului, atributele și stările lor locale, care pot să le primească;

– condițiile și evenimentele ce pot schimba stările, fie că ele provin de la un proces aleatoriu, fie de la o decizie specificată;

– condițiile de sincronizare și cooperare, care determină apariTia unor evenimente;

– atributele calitative și cantitative ce determină restricțiile de funcționare ale sistemului;

– specificațiile condițiilor interacțiunii evenimentelor şi nivelul de detaliere al modelării, redate de o descriere informală a funcționării sistemului considerat.

Îndată ce aceste elemente diferite sunt identificate, folosind atributele specificate, o descriere informală a proceselor interacțiunii lor şi un *raționament adecvat*, este posibil de a compune o expresie descriptivă a unei rețele *GeN*, ce constituie o descriere logică a comportamentului sistemului de calcul considerat.

În [3] este arătat cum se poate efectua maparea DE a unei *GeN* în reprezentare grafică a acestei rețele şi invers.

Pentru a efectua analiza unor modele GSPN, redate prin DE, a fost elaborat şi implementat un subsistem program de scriere şi redactare a codului ce redau DE respective, care a fost integrat în platforma PIPE v4.30, deoarece codul sursă, scris în limbajul Java, poate fi accesat liber [6].

III. PLATFORMA PIPE

Platforma - Independent Petri Net Editor (PIPE) Open-Source [6] este un instrument program bazat pe limbajul Java pentru construcția şi analiza RP stocastice generalizate (GSPN). Pe lângă crearea modelelor de rețele Petri standard, manipulare şi facilități de animație, PIPE prevede un mecanism pentru integrarea run-time a noi funcționalități printr-un modul pluggable de analiza. Aceasta este o caracteristică care evidențiază PIPE de multe alte instrumente de simulare a RP, a căror funcționalităȚi de analiză sunt, de obicei, fixate şi nu pot fi extinse de către utilizator. PIPE oferă, prin urmare, o rampă de lansare pentru experimentare cu noi tehnici de analiză, fără necesitatea de a reimplementa funcționalitățile de bază. De asemenea, PIPE oferă un instrument intuitiv, uşor de utilizat în editarea RP într-un mod simplu, rapid şi eficient. Orice persoană familiarizată cu un alt editor graphic utilizator (GUI) pentru RP, se poate acomoda simplu cu interfața oferită de PIPE, fără necesitatea de a avea cunoştințe suplimentare. GUI al PIPE foloseşte reprezentarea standard a RP, iar meniul cu bara de instrumente poate fi înțeles şi învățat foarte uşor.

PIPE oferă şi un mod de animare destul de comod, astfel încât utilizatorul poate experimenta în mod manual cu animația jetoanelor, pas cu pas, urmărind fiecare activare de tranziție, la fiecare pas. Setul de tranziții activate sunt evidențiate cu culoare roşie. Istoricul animației este înregistrat şi există posibilitatea de a se întoarce la un pas de simulare precedent, şi respectiv există posibilitatea de mers înainte, până la ultimul pas simulat. Este posibilă executarea aleatoare a tranzițiilor, pentru aceasta utilizatorul trebuie să specifice ratele de declanşare ale acestora. Al doilea mod de simulare, este automat. În acest regim utilizatorul specifica numărul de paşi care vor fi executați în mod automat. De asemenea, PIPE oferă un set de module pentru a efectua diferite tipuri de analize calitative şi cantitative. Acest set de module poate fi uşor extins prin adăugarea a unor module proprii, cu condiția ca să fie implement accesul la ele prin GUI al PIPE.

IV. SUBSISTEMULUI DE COMPUNERE A REȚELELOR PETRI ÎN PIPE

Aplicația dată a fost elaborată în Eclipse, pentru implementarea acestui subsistem a fost utilizat limbajul de programare Java.

Pentru crearea şi editarea modelelor RP redate prin DE a fost elaborat şi implementat un editor special care poate fi accesat din meniul GUI al PIPE .

Dacă avem o RP creată în fereastra activă a GUI atunci, la rularea programului (editorului de text accesat prin-un butonul selectat în Fig. 2), se va crea un fişier temporar cu extensia *.fl* în care vor fi înscrise DE ale rețelei curente. Dacă fereastra lipseşte sau fereastra este activă, dar ea e vidă, atunci editorul de text se va deschide şi nu se va crea nici un fişier al DE.

Editorul de text creat este simplu şi uşor de lucrat cu el şi posedă un set minimum de comenzi necesare. Interfața grafică a acestui meniu este alcătuită din 3 elemente principale: Meniul principal (I, II); Zona de text (III, IV); Mesaje (V) (vezi Fig. 2). Semnificația acestor elemente este: I. *Bara de acces rapid*; II. *Conține comenzile accesate cel mai des*; III. *Suprafața de text unde este scris codul*; IV. *Numărul rândului în text* ; V. *Zona de mesaje în care sunt afişate erorile depistate în urma compilării codului.* Meniul principal al editorului de text DE este alcătuit din următoarele elemente:

- *File* - comenzile de manipulare cu fişierul: *New*; *Open*; *Save*; *Save as…* ; *Exit.*
- *Edit* - comenzile de manipulare cu textul: *Undo; Redo; Copy; Paste; Cut; Delete; Select all.*
- *Source* - la accesarea căruia, se inserează un *template* al unei RP care, poate fi editat în continuare;
- *Run* - rulează compilarea codului;

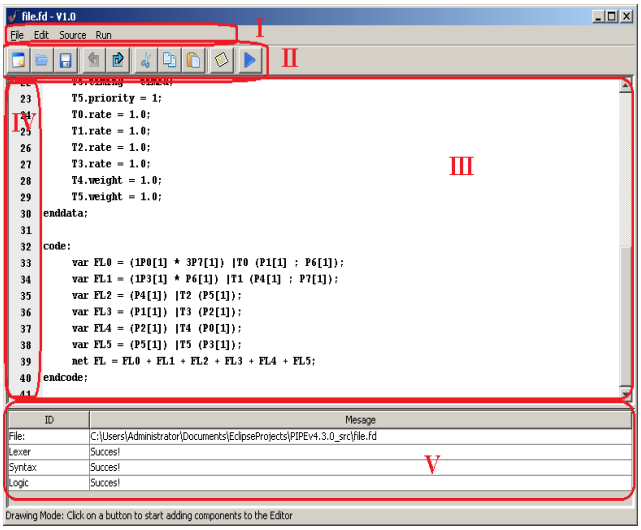

Fig. 2. Componentele editorului de text al DE.

Pentru a crea şi edita o RP, redată de o DE, se foloseşte un editor special care se accesează din meniul PIPE.

Listing-ul textului va fi împărțit în 2 blocuri: *data* – pentru declararea locațiilor şi a tranzițiilor; *code* – pentru descrierea prin formule a subrețelelor RP.

În blocul *data*, după cuvântul cheie *tranzition*, declarăm tranzițiile, cu nume unice, ce vor fi atribuite ca ID pentru aceste tranziții. În mod similar se vor declara şi locațiile - *place*. În continuare, în acest bloc dat se vor seta valorile pentru locațiile şi tranzițiile declarate. Acest procedeu se va face prin menționarea numelui nodului declarat anteperior, urmat de simbolul special „*punct*”, după care urmează numele pentru valoare căruia el se setează.

Pentru locații, avem următoarea sintaxă: de exemplu *p1.capacity* = 10; În cazul în care nu se scrie rândul dat, locația declarată va primi în mod implicit capacitatea infinită, în program ea este reprezentată ca fiind 0.

Fiecare rând de cod se termină cu simbolul special "**;**", după el pot urma comentarii până la următorul rând nou.

Pentru tranziții:

- *t1.server = simple; // sau multiple*;
- *t1.timing = timed; // sau immediate*;
- *t1.priority = 123*;
- *t2.rate* = 5.

În funcție de tipul tranziției, pentru atributele *speed* sau *rate* poate fi scrisă formula DE. Astfel daca tranziția este imediată, se scrie *t.weigth*, în cazul în care tranziția este temporizată, se scrie *t.rate*, după cum este exemplificat:

- *t1.weight = 1.0+5-#(p1)+cap(p0)+exp(#(p1)*cap(p0))*;
- *t2.rate = 1.0+5-#(p1)+cap(p0)+exp(#(p1)*cap(p0))*.

În cazul în care lipsesc unele declarații ale tranzițiilor, atunci lor vor fi asignate valori implicite, de exemplu:

t.server = simple; *t.timing = immediate*;
automat t.priority = 1; *t.rate* = 1; *t. weight* = 1.

În cazul în care există erori în codul scris, la compilarea lui, rețeaua RP nu va fi desenată şi se va afişa un mesaj sugestiv cu indicația liniei unde a fost depistată eroarea.

În cazul în care apar ceva probleme referitoare la sintaxă textului, poate fi accesat modelul din meniul source, care reprezintă un *template*.

În blocul *code* se scriu formulele DE propriu zise. Fiecare subrețea se declară prin cuvântul cheie *var* urmat de numele care se asociază ei. După semnul egal se scrie formula DE ce descrie subrețeaua dată. Pentru formarea rețelei integrale care va fi desenată, se scrie cuvântul cheie *net* urmat de un nume asociat. După semnul *egal* se scrie semnul + ce indică aplicare

operaȚia *Operația Paralelism competitiv reuniunea* subrețelelor care vor forma rețeaua finală, după cum poate fi urmărit următorul exemplu:

code:
var FL3 = (p4[1] * p3[1]) |t5 (p2[7]);
var FL4 = (p5[1] * p4[1]) |t6 (p7[2+#(p6), *cap*(p4)]
|t7 p8[1]; p6[1]; p1[14]);
net RP1 = FL1 + FL2;
net RP2 = FL3 + FL4;
net RP3 = FL1 + FL2 + FL3 + FL4;
endcode;

Acest editor are o interfață asemănătoare cu interfața de lucru a aplicației principale. Din meniul *file* putem crea un fişier nou sau deschide unul existent. În acelaşi mod, putem salva sau închide pagina curentă.

Din meniul „*Edit*" avem posibilitatea de a merge înapoi sau înainte prin istoricul editărilor. De asemenea, avem posibilitatea de a tăia o porțiune de cod, de a copia sau insera porțiunea de cod dată. La fel putem şterge porțiunea de cod selectată şi în plus, putem selecta tot codul odată. Pentru fiecare operație există o combinație de taste scurte, care uşurează semnificativ lucrul.

Din meniul "*Source*" putem deschide o pagină nouă cu un cod sursă de probă, care poate fi ulterior modificat, fiind ca un punct de reper.

Compilarea codului formulelor DE se efectuează din editorul de text prin tastarea butonului *Run* sau *F11* care va apela metoda *actionPerformed(e:ActionEvent)*. Diagrama de secvențe a compilării codului DE este prezentată în Fig. 3.

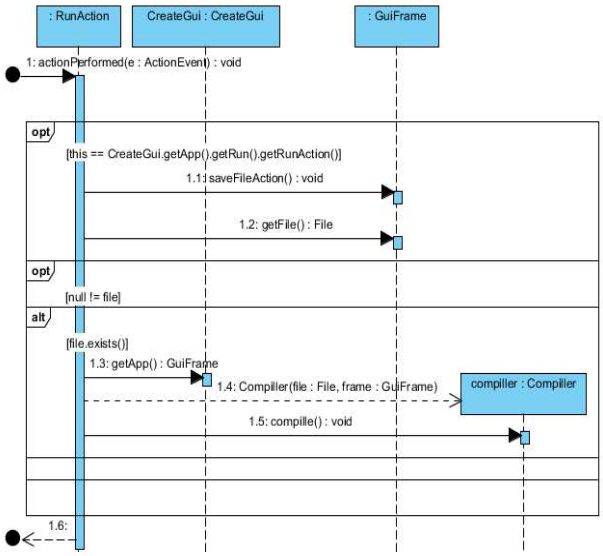

Fig. 3. Diagrama de secvențe a compilării codului DE în PIPE.

Din meniul "*Run*" putem rula programul. În cazul în care nu există erori, se va afişa un mesaj de confirmare a corectitudinii codului scris şi se va desena RP descrisă de codul dat. În cazul în care există erori, se va afişa mesajul respectiv care confirmă existența erorii şi locul ei.

La crearea grafică a unei RP redată prin DE nu este rațional de a specifica coordonatele *nodurilor* care urmează a fi conectate între ele, deoarece acest procedeu ar complica semnificativ structura formulelor. În acest scop au fost elaborați doi algoritmi de desenare automată a RP, redată de DE în formă textuală sau la redesenarea unei rețele existente. La selectarea butonului de desenare, rețeaua va fi redesenată în fereastra curentă. Pentru comoditate exista combinații de taste pentru aplicarea acestui algoritm în fereastra curentă, combinația scurtă este *ctrl + shift + E* (vezi Fig. 4).

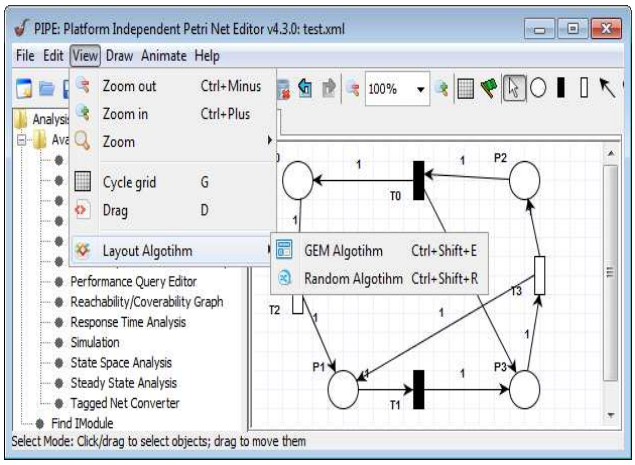

Fig. 4. Meniul de selecție a algoritmului de redesenare a RP redată prin DE.

Diagrama de activitĂȚi a algoritmului de poziȚionare a nodurilor (locaȚii şi/sau tranziȚii) ale RP este reprezentată Fig. 5.

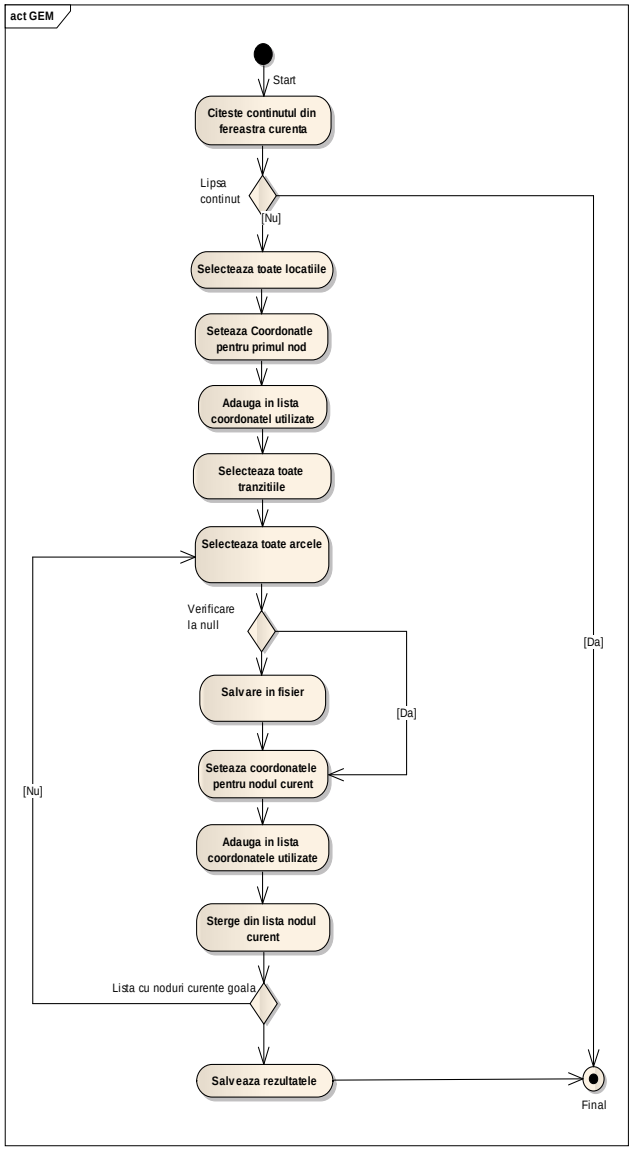

Fig. 5. Diagrama de activităţi a algortmului de pozitionare grafică a RP redată prin DE.

Subsistemul elaborat este implementat şi integrat în PIPE v4.30. El oferă utilizatorului următoarele noi funcţionalităţi:

- a deschide o reţea nouă în fereastra de lucru, fie dintr-un fişier salvat în codul XML, care descrie structura reţelei RP, fie dintr-un fişier salvat în format textual al DE.
- a aplica un algoritm de aranjare automată a reţelei din fereastra curentă, fie aplicând algoritmul de aranjare aleatoare a locaţiilor şi tranziţiilor, fie aplicând algoritmul de aranjare specială pentru aranjarea reţelei după o logică stabilită.
- are posibilitatea de a trece din regimul grafic în cel analitic, redată de DE şi invers.
- dacă în fereastra curentă este desenată o reţea, la tastarea butonului special pentru deschiderea editorului de formule DE, se va deschide fereastra de lucru cu aceste formule pentru reţeaua dată.
- în cazul în care ne aflăm în fereastra editorului de formule, după scrierea formulelor, putem compila acest cod pentru verificarea prezenţei unor erori. Dacă acestea nu există, se va desena RP respectivă, fiind mapată în mod automat de către DE descrisă în editorul de formule. RP astfel creată, cu toate atributele sale, se va memora într-un fişier cu extensia *.xml* [7] care, ulterior, va fi deschis în PIPE v4.3.0 şi apoi analizat, folosint toate aplicaţiile sale.

MenŢionăm că în cadrul acestui subsistem putem crea o bibliotecă de subrŢele RP "*şablon*" în formă de DE şi apoi le putem folosi, la necesitate, pentru a compune modele reale ale procelor de calcul.

VI. CONCLUSII

În lucrare sunt considerate unele aspecte de elaborare şi implementare în limbajul Java a unui subsistem program pentru compunerea modelelor de reţele Petri stocastice generalizate prin expresii descriptive care permit de a formaliza etapa de trecere logică de la o descriere informală a arhitecturii şi a specificaţiilor comportamentale ale sistemului analizat la maparea lor în modele GSPN. Acest subsistem este integrat în mediul PIPE v4.30 (Independent Petri Net Editor Open-Source) [6], care pe lângă crearea redactarea, simularea, analiza comporta-mentală şi a facilităţilor de animaţie a modelelor GSPN, el prevede şi un mecanism de integrare run-time a noi funcţionalităţi printr-un modul pluggable de analiză.

Pe viitor se prevede de a introduce noi module program în mediul PIPE v4.30 modificat cu noi facilităŢi, care vor permite de a efectua simularea vizuală, analiza comportamentală şi evaluarea performanŢelor modelelor GSPN ale proceselor de calcul reconfigurabile [5].

Lucrarea dată a fost efectuată în cadrul proiectelor naŢionale de cercetări ştiinŢifice aplicative 11.817.08.55A *şi* 14.820.18.02.03/U.

REFERENŢE

[1] M. Ajmone-Marsan, G. Balbo, G. Conte. "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.

[2] E. Guţuleac. *Evaluarea performanţelor sistemelor de calcul prin reţele Petri stochastice*. Editura „Tehnica-Info", Chişinău, 2004, - 276 p., ISBN 9975-63-228-9.

[3] E. Guţuleac. "Descriptive Compositional Construction of Generalized Stochastic Petri Net Models for Performance Evaluation of Computer Systems," Buletinul Institutului Politehnic din Iaşi, Tomul L (LIV), Fasc. 1-4, Secţia Automatică şi Calculatoare, 2004, p. 143-159, ISSN 1220-2169.J.

[4] E. Guţuleac. "Descriptive compositional HSPN modeling of computer systems," Annals of the University of Craiova, România, Series: Atomation, Computers, Electronics and Mechatronics, Vol. 3(30), No.2, 2006, p. 82-87, ISSN 1841-0626.

[5] Iu Ţurcanu, E. Guţuleac. "Modelarea sistemelor orientate pe servicii prin reţele Petri reconfigurabile cu atribute matriceale," Meridian ingineresc, Nr. 1, 2014, p.31-38, Ed.: UTM, ISSN 1683-853X.

[6] PIPE2: Platform-Independent Petri net Editor, Available: http://pipe2.sourceforge.net.

[7] The Petri Net Markup Language, Available: http://www2.informatik.hu-berlin.de/top/pnml/.