DIAGRAMELE ESC ÎN PRISMA SEMANTICII REȚELELOR YASPER (PETRI)

D. Ciorbă, drd
Universitatea Tehnică a Moldovei

INTRODUCERE

Necesitatea de analiză operațională a diagramelor ESC (Event-based Specification Chart, [1]) îndreptă direcția de cercetare spre rețelele Petri, ca fiind un mijloc unic în perspectiva corespondenței structurale. Din multitudinea de instrumente prezentate în [2], principiul multicriterial (arcuri de resetare şi inhibiție, noduri decizionale, subrețele, fluxuri de jetoane specifice utilizate la modelarea de procese *workflow*, în care un rol important joacă conceptul *caz*) potrivește *Yasper* în alegerea instrumentului de analiză.

Un *caz* este o entitate care poate fi tratată ca o instanță de proces şi are o *istorie* reflectată prin sarcini (modelate prin tranziții asociate unor roluri).

Prin urmare *un* proces *Yasper* poate fi reprezentat de o rețea Petri (structural - un graf), care posedă un singur nod de început – *emițător de jetoane*, un singur nod final – *colectorul jetoanelor*, alte noduri care determină traseul de execuție. Într-un proces pot fi active mai multe cazuri şi acestea trebuie considerate independente unu de altul. Dar dacă se dorește interacțiunea cazurilor, se poate recurge la utilizarea de poziții şi jetoane independente de caz (*uncased* – eng.*)*. [3]

1. REȚELE PETRI

Modelarea comportamentului în specificațiile ESC este centrată pe evenimente şi cauzalitate[1]. Prin urmare o rețea Petri va fi *interpretată* ca un tuplu $N= \langle B, E, F, M \rangle$, unde: B este mulțimea condițiilor (tradițional reprezentate prin poziții); E este mulțimea evenimentelor (tradițional reprezentate prin tranziții); *şi* $F \subseteq (B \times E) \cup (E \times B)$ este relația de dependență cauzală, cu: $B \cap E \neq \varnothing$.

Pentru $x \in B \cup E$, se va nota prin $^\bullet x$ (respectiv $x^\bullet$) expresia de tip $\{ y \mid yFx \}$ (respectiv expresia de tip $\{ y \mid xFy \}$). Prin urmare dacă $x \in E$, atunci $^\bullet x$ defineşte o mulțime de *precondiții* şi $x^\bullet$ defineşte o mulțime de *postcondiții*; iar dacă $x \in B$, atunci $^\bullet x$ defineşte o mulțime de *preevenimente* şi $x^\bullet$ defineşte o mulțime de *postevenimente*.

1.1. Rețele de ocurență

O clasă deosebită de rețele Petri formează *rețelele de ocurență* (*occurrence net* – eng.), care sunt rețele satisfăcând următoarele cerințe [4, p. 104, 5, p. 368]:

- $\forall\ b \in M,\ |^\bullet b| = 0,$ marcajul inițial este determinat de condițiile care nu sunt precedate de nici un eveniment (prin relația *F*);
- $\forall b \in B,\ |^\bullet b| \leq 1,$ orice condiție este precedată de maximum un eveniment*;*
- F^+ este o închidere antireflexivă pentru *F;* nefiind permise cicluri în dependențele induse din *F,* se notează cu F^* şi implică o mulțime finită*:* $\forall e \in E, \{ e' \mid e' F^* e \}$ *[4, p. 38];*
- # este o relație antireflexivă de conflict: $x \# x' \Leftrightarrow_{def} \exists\ e, e' \in E.\ e \#_m e'$ şi eF^*x şi $e'F^*x'$, unde $(\#_m)$ - este relația de conflict imediat: $e \#_m e' \Leftrightarrow_{def} e \in E$ şi $e' \in E$ şi $^\bullet e \cap {}^\bullet e' \neq 0$.

Ultima condiție este absolut necesară: ea elimină evenimentele care nu pot apărea în aceeaşi „execuție”, deoarece depind de apariția evenimentelor conflictuale precedente.

Conform teoremelor prezentate în [4, p. 110] o structură de eveniment *E* poate induce o rețea de ocurență $N(E) = \langle B, E, F, MI \rangle$, care poate fi construită după cum urmează:

- $MI = \{ (e, e') \mid e, e' \in E\ \ e \#_m e' \};$
- $B = \{ (e, e') \mid e < e' \}$, deci $B \subseteq E^2;$
- $F = \{ (e, (e, e')) \mid e, e' \in E$ şi $(e, e') \in B \} \cup \{ ((e, e'), e') \mid e, e' \in E$ şi $(e, e') \in B \} \cup \{ ((e, e'), e) \mid e, e' \in E$ şi $(e, e') \in MI \} \cup \{ ((e, e'), e') \mid e, e' \in E$ şi $(e, e') \in MI \};$

Prin urmare, dacă o structură de evenimente *ES* este reprezentată de evenimentele, dependențele cauzale şi conflictul din figura 1-a, atunci acestei structuri i se poate pune în corespondență rețeaua de ocurență din figura 1-b.

1.2. Rețele de tranziții

În [4, p. 22] se menționează că o rețea Petri poate fi descrisă în câteva nivele conceptuale. Un

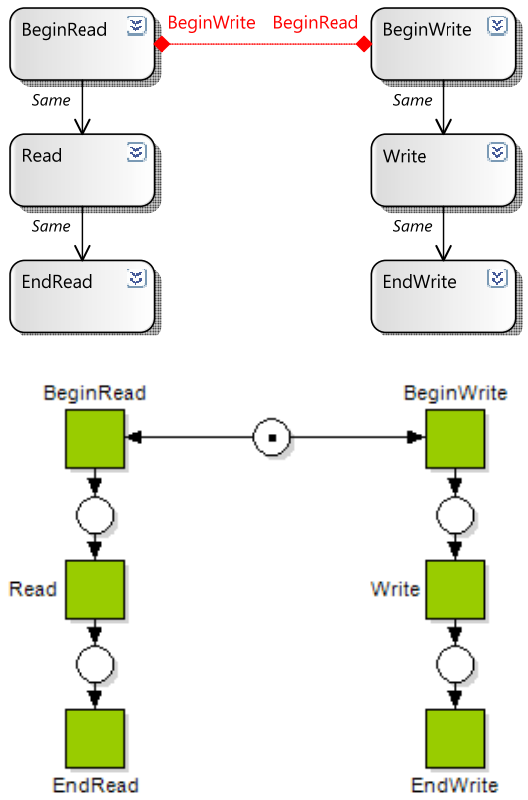

Figura 1. Structura de evenimente *ES = (E ≤, #)* (a) induce rețeaua de ocurență *N(ES) = ⟨B, E, F, MI⟩ (b)*

nivel conceptual important este cel al *tranzițiilor,* care adaugă unei topologii comportament dinamic prin *marcajul* ce se modifică la declanşarea evenimentelor. Se consideră că un eveniment *e* obține *concesia* (pentru declanşare) în cazul unui marcaj *M*, dacă: $^\bullet e \subseteq M$ şi $e^\bullet \cap M = \varnothing$. Prin urmare *relația de tranziție* dintre două marcaje *M* şi *M′*, notată prin $M\ [X\rangle\ M'$ sau $M \overset{X}{\Longrightarrow} M'$, devine declanşatoare pentru $X \subseteq E$ dacă: a) orice membru ce aparține X are concesia la marcajul M; b) nici un membru ce aparține X nu este în conflict la marcajul M; c) $M' = M - \cup\{ {}^\bullet e \mid e \in X \} + \cup\{ e^\bullet \mid e \in X \}$.

2. INTERPRETAREA SPECIFICAȚIILOR ESC

În figura 2 se observă că specificațiile ESC adaugă un nivel conceptual asupra structurilor de evenimente prin: a) specificarea rolurilor incidente evenimentelor; b) gruparea (localizarea) evenimentelor în nori; c) specificarea numărului maxim de instanțe de rol active în nori, indicând multiplicitatea norilor mai mare sau egală cu unu; d) etichetarea dependențelor cu două atribute (ocurență de eveniment şi incidență de rol); etc. Fiecare *rol* prezent în diagramă defineşte un comportament comun pentru toate instanțele lui. Transpunerea conceptului de *instanță de rol* poate fi realizată în rețele *Yasper* utilizând conceptul de *caz*, care modelează o instanță de proces Yasper.

Relevante întru înțelegerea conceptului de caz sunt exemplele următoare: a) dacă un eveniment

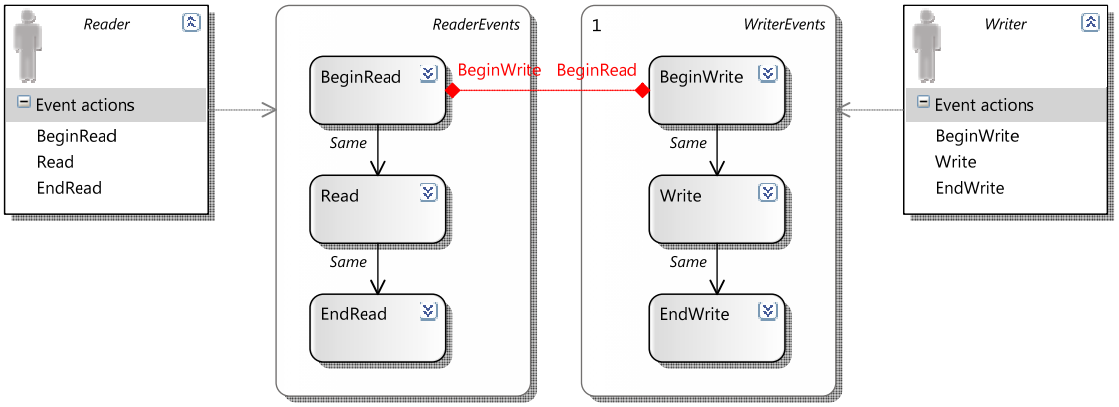

Figura 2. Evenimente localizate în nori şi roluri incidente specificate în diagrame ESC

Un *caz* se manifestă prin jetoane generate de un emițător, care le „imprimă" identitate. Astfel ne fiind obiecte anonime la mişcarea prin rețea, jetoanele de *caz* poartă o informație ce facilitează determinarea dependențelor cauzale dintre evenimente (tranziții).

are două precondiții (în mod evident, sunt două poziții *de caz*), atunci declanşarea acestui eveniment va fi posibilă doar dacă în ambele poziții vor fi jetoane ale aceluiaşi caz (figura 3-a); b) dacă într-o poziție sunt două jetoane de caz, evenimentul se va declanşa separat pentru fiecare (figura 3-b).

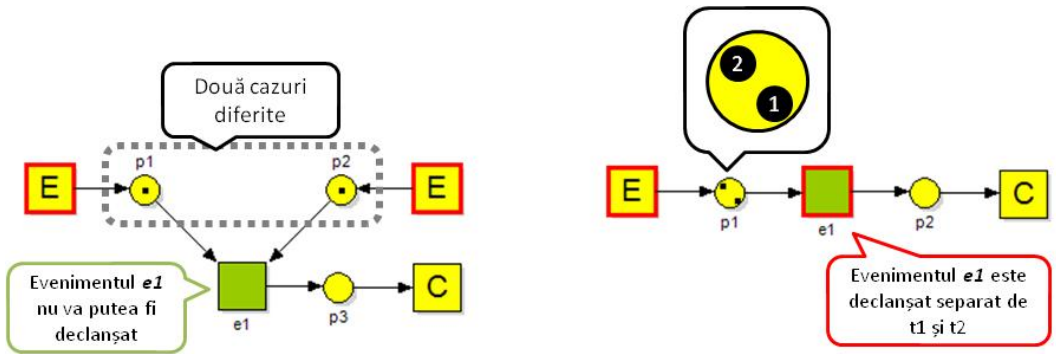

Figura 3. Declanşarea evenimentului este determinată de caz

Cele relatate mai sus impune redefinirea (reformularea) marcajului şi relației de tranziție dintre marcaje întru interpretarea corectă a specificațiilor generate. În această ordine de idei este util conceptul *traseelor decorate* enunțate în [6] şi aplicabil rețelelor Petri de forma $N = \langle S, T, F, M_0 \rangle$ şi constă în etichetarea (*labelling* – eng.) tuturor jetoanelor în rețea:

- Prin $T^* \triangleq T \cup \mathbb{N}$ se notează mulțimea etichetelor, astfel că un marcaj M de jetoane etichetate în rețeaua N este definit prin $M: S \rightarrow \Lambda(T^*)$, unde Λ este relația de etichetare;
- O tranziție (t,V) se consideră validă (enabled – *eng.)* la marcajul etichetat M, notând $M \overset{(t,V)}{\Longrightarrow}$, dacă $V: {}^\bullet t \rightarrow T^*$, astfel încât $\forall s \in {}^\bullet t: V(s) \in M(s)$; Aşadar, dacă în poziția s sunt două jetoane, fie t_1 şi t_2, atunci $(t, (s, t_1))$ şi $(t, (s, t_2))$ reprezentă validări diferite ale tranziției t;
- Prin urmare modificarea marcajului în rețea poate fi reprezentată prin:

$$M'(s) = \begin{cases} M(s) - V(s) \cup \{t\}, & \text{dacă } s \in {}^\bullet t \cap t^\bullet \\ M(s) - V(s), & \text{dacă } s \in {}^\bullet t \wedge s \notin t^\bullet \\ M(s) \cup \{t\}, & \text{dacă } s \notin {}^\bullet t \wedge s \in t^\bullet \\ M(s), & \text{în caz contrar.} \end{cases}$$

Relația de tranziție dintre marcaje este determinată inclusiv de alte elemente conceptuale ale specificației ESC. Printre care un rol important îl are *norul de evenimente*, care a fost introdus în diagramele ESC întru îmbunătățirea reutilizării. Rafinamentele aplicate norilor în acest scop nicidecum nu influențează domeniul semantic al specificațiilor, ci doar sintactic, definind un *homomorfism* [7, 8]. Prin urmare interpretarea diagramelor şi transpunerea acestora în rețele ar trebui realizată doar după ce este realizat rafinamentul. În pofida acestui fapt, dacă limbajul țintă oferă o structurare corespunzătoare în domeniul său sintactic, aceasta poate fi aplicată pentru a utiliza pe deplin facilitățile limbajului.

Rețelele Petri create cu ajutorul instrumentului Yasper pot fi structurate ierarhic în subrețele, care corespund conceptual *norului de substituție eveniment* ale diagramelor ESC. Rafinamentul prin *superpoziție nor* nu are o corespondență sintactică în rețele Petri şi va fi realizat până la transpunere prin generarea dependențelor induse de relațiile de *specializare* a evenimentelor.

Deoarece norul de evenimente are şi un rol de mărginire a proprietății conflictului ereditar (proprietate a unei structuri de evenimente) în rețea trebuie introduse poziții şi relații noi ce ar invalida evenimentele aflate în *conflict imediat* până la *ocurența* evenimentelor terminale din nori:

- $B_c = \left\{ (c1, (e1, e2))_u,\ (c2, (e1, e2))_u \mid (e1, e2)_u \in MI \wedge e1 \in c1 \wedge e2 \in c2 \wedge c1, c2 \in C \right\}$, unde C este o mulțime a norilor, ce includ evenimente ordonate cauzal, iar pozițiile mulțimii sunt de tip *uncased,* fiind marcate prin indicele u;
- $F_c = \{(e1, (e1, e2)_u), (e2, (e1, e2)_u) \mid (e1, e2)_u \in MI\} \cup$
  $\left\{ \left(e1, (c1, (e1, e2))_u\right), \left(e2, (c2, (e1, e2))_u\right) \middle| (c1, (e1, e2))_u, (c2, (e1, e2))_u \in B_c \wedge (e1, e2)_u \in MI \wedge e1 \in c1 \wedge e2 \in c2 \wedge c1, c2 \in C \right\} \cup \cup \left\{ \left((c, (e1, e2))_u, e\right) \mid e \in \lfloor c \rfloor, (c, (e1, e2))_u \in B_c \right\}$,
  unde $\lfloor c \rfloor$ este mulțimea evenimentelor ce mărginesc inferior norul c;
- $I_c = \left\{ \left(e1, (c2, (e1, e2))_u\right), \left(e2, (c1, (e1, e2))_u\right) \middle| (c1, (e1, e2))_u, (c2, (e1, e2))_u \in B_c \wedge (e1, e2)_u \in MI \wedge e1 \in c1 \wedge e2 \in c2 \wedge c1, c2 \in C \right\}$, prin I_c vom nota mulțimea arcelor de inhibiție;
- $W_c = \left\{ \left(\left(e1, (c1, (e1, e2))_u\right), ||\lfloor c1 \rfloor|| \right), \left(\left(e2, (c2, (e1, e2))_u\right), ||\lfloor c2 \rfloor|| \right) \right\}$

Multiplicitatea norului implică adăugarea a două evenimente auxiliare (de intrare şi ieşire din nor), care trebuie conectate obligatoriu prin intermediul unor poziții de caz cu evenimentele ce mărginesc superior şi, respectiv, inferior norul. Evenimentele introduse se interconectează reciproc prin intermediul unei poziții independente de caz (*uncased* – eng.), căci este destinată pentru a asigura interacțiunea dintre cazuri şi are menirea de a indica numărul maxim de instanțe active ale rolurilor incidente norului:

- $E_m = \{e_{in}, e_{out} | \ \mathcal{M}(c) \geq 1\}$, unde $\mathcal{M}(c)$ este multiplicitatea norului *c;*
- $B_m = \cup\{(e_{in}, e)_c \ | e_{in} \in E_m \wedge e \in \lceil c \rceil \wedge \mathcal{M}(c) \geq 1\} \cup \ \cup\{(e, e_{out})_c \mid e_{out} \in E_m \wedge e \in \lfloor c \rfloor \wedge \mathcal{M}(c) \geq 1\};$
- $MI_m = \{((e_{in}, e_{out})_u, \mathcal{M}(c)) | \ e_{in}, e_{out} \in E_m \wedge \ e_{in}, e_{out} \in c\};$
- $F_m =$
  $\{(e_{out}, (e_{in}, e_{out})_u), ((e_{in}, e_{out})_u, e_{in}) \mid e_{in}, e_{out} \in E_m \wedge (e_{in}, e_{out})_u \in B_m\} \cup$
  $\cup\{(e_{in}, (e_{in}, e)_c), ((e_{in}, e)_c, e)| \ (e_{in}, e)_c \in B_m \wedge e \in \lceil c \rceil \wedge \mathcal{M}(c) \geq 1\} \cup$
  $\cup\{(e, (e, e_{out})_c), ((e, e_{out})_c, e_{out})| \ (e, e_{out})_c \in B_m \wedge e \in \lceil c \rceil \wedge \mathcal{M}(c) \geq 1\}$

Prin urmare o rețea Petri/Yasper ce ar corespunde specificației ESC din figura 2, fiind inițial determinată de rețeaua de ocurență $N = \langle B, E, F, MI \rangle$ obținută din structura de evenimente a specificației, are forma $N_{esc} = \langle B_{esc}, E_{esc}, F_{esc}, MI_{esc}, I_{esc}, W_{esc} \rangle$ şi se prezintă în figura 4, unde:

- $B_{esc} = B \cup B_c \cup B_m = \{r1_c, r2_c, r3_c, r4_c, w1_c, w3_c, w4_c, w6_c\} \cup \{ keepConflict1_u, keepConflict2_u\} \cup \{w2m_c, \ w5m_c, multiplicity_u\};$
- $E_{esc} = E \cup E_m =$
  $\{BeginRead, \ Read, \ EndRead, \ BeginWrite, \ Write, \ EndWrite\} \cup \{SelectWriter, \ OutWriter\};$
- $F_{esc} = F \cup F_c \cup F_m =$
  $F \cup$
  $\{(BeginRead, \ co \ nflict_u),$
  BeginWrite, conflictu, BeginRead, keepConflict1u, BeginWrite, keepConflict2u, keepConflict1u, EndRead, keepConflict2u,EndWrite $\cup$ *OutWriter, multiplicityu, multiplicityu,SelectWriter, SelectWriter,w2mc, w2mc,BeginWrite, EndWrite, w5mc,w5mc,OutWriter;*

- $MI_{esc} = MI \cup MI_m = \{(conflict_u, 1)\} \cup \{(multiplicity_u, 1)\};$
- $I_{esc} = I_c = \{(BeginRead, keepConflict2_u), (BeginWrite, keepConflict1_u)\};$
- $W_{esc} = W_c = \{((BeginRead, \ keepConflict1_u), 1), ((BeginWrite, keepConflict2_u), 1)\};$

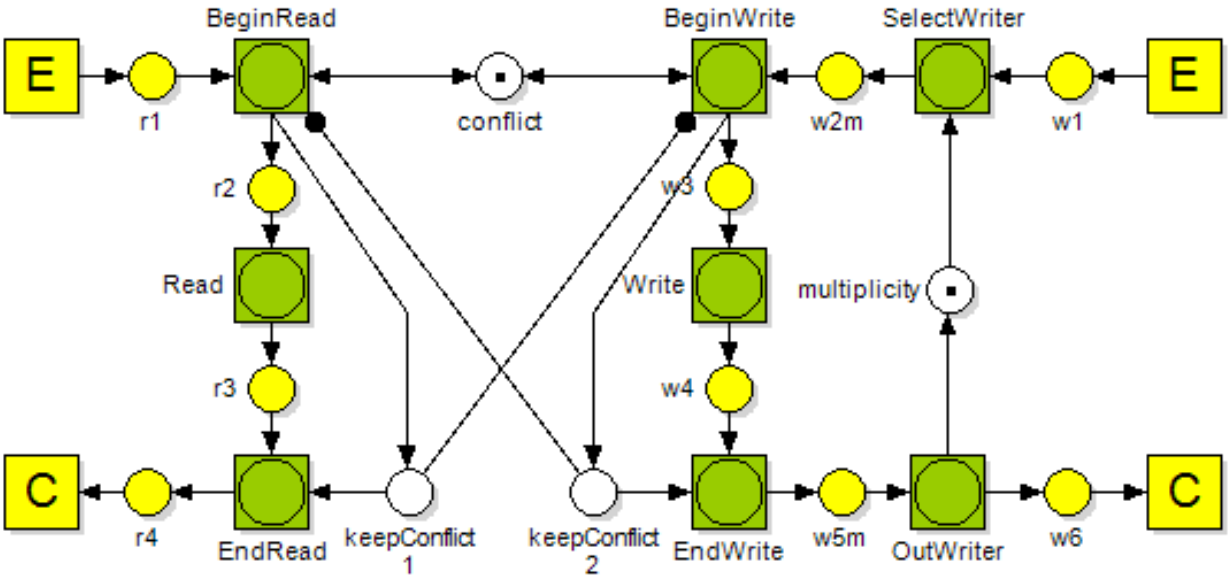

Figura 4. Rețea Petri/Yasper colorată şi cu arcuri inhibitor

CONCLUZII

Alegerea instrumentului Yasper este justificată, pe lângă corespondența structurală ale rețelelor sale cu diagramele ESC, şi prin editarea uşoară a modelelor, animarea jetoanelor la simulare, analiza primară a performanțelor cu o simulare automată aleatorie a rețelelor. Obținerea acestora din diagramele ESC este bazată pe ideea nivelelor conceptuale în descrierea rețelelor Petri, menționată în [4, p. 17]. Acest fapt, deşi este unul semi-formal, facilitează înțelegerea transformării graduale a unei specificații ESC într-o rețea Yasper, rețea ce cu jetoanele sale *case-sensitive* poate fi considerată o aplicare limitată şi specifică a rețelelor Petri colorate cu arcuri inhibitor.

Bibliografie

1. ***Ciorba, D., Besliu, V.*** *Concurrency specification using Event-based Specification Chart. Computer Science Journal of Moldova. Chisinau : Institute of Mathematics and Computer Science (Academy of Sciences of Moldova), 2011. Vol. 19, 3 (57), pg. 231-253. http://www.math.md/publications/csjm/issues/v19-n3/10998/. ISSN 1561-4042.*

2. ***Heitmann, F., Moldt, D.*** *Petri Nets Tool Database. Petri Nets World. [Interactiv] 06 06 2011. [Citat: 07 05 2012.] http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db.html.*

3. ***van Hee, Kees, ş.a.*** *History-Based Joins: Semantics, Soundness and Implementation. [ed.] Schahram Dustdar, José Luiz Fiadeiro şi Amit Sheth. Business Process Management: 4th International Conference, BPM 2006: Proceedings. 5 Octomber 2006, pg. 225-240. http://books.google.md/books?id=3Yh9bdxeW1UC&pg=PA225.*

4. ***Winskel, G.*** *Events in computation. University of Edinburg. 1980. PhD thesis. http://www.daimi.au.dk/~gwinskel/Events-in-Computation.pdf.*

5. —. Event structures. [ed.] Wilfried Brauer, Wolfgang Reisig şi Grzegorz Rozenberg. Petri Nets: Applications and Relationships to Other Models of Concurrency. Berlin : Springer-Verlag, 1987, Vol. 255, pg. 325-392. http://www.cl.cam.ac.uk/~gw104/EvStr.pdf.

6. ***Katoen, J.-P.*** *Causal behaviors and Nets. [ed.] Giorgio De Michelis şi Michel Diaz. Application and theory of Petri Nets. s.l. : Springer, 1995, Vol. 935, pg. 258-277. http://doc.utwente.nl/66278/1/269_atpn95.pdf.*

7. ***Goltz, U., Gorrieri, R., Rensink, A.*** *Comparing Syntactic and Semantic Action Refinement. information and computation. s.l. : Academic Press, Inc., 1996. 125, pg. 118-143.*

8. ***Gorrieri, R., Rensink, A.*** *Action refinement. [ed.] Jan Bergstra, Alban Ponse şi Scott Smolka. Handbook of Process Algebra. s.l. : Elsevier, 2000.*

Recomandat spre publicare: 24.07.2013.