Aspecte de Modelare şi Analiză a Proceselor de Calcul Orientate pe Servicii Reconfigurabile

Iurie ŢURCANU, Emilian GUŢULEAC, Diana PALII
Technical University of Moldova
egutuleac@mail.utm.md

***Abstract* — În lucrare sunt considerate unele aspecte arhitecturale, de modelare şi analiză a proceselor sistemelor de calcul orientate pe servicii reconfigurabile (*OSR*) prin reţele Petri reconfigurabile stocastice (*RPRS*). În acest context este prezentată o metodă de formalizare a etapei de trecere logică de la o descriere informală a arhitecturii şi a specificaţiilor comportamentale ale sistemului analizat la determinarea unor expresii descriptive şi maparea lor în modele de reţele *RPRS* ale acestor procese, subiacente modelelor de reţele Petri stocastice. Această abordare permite de a diminua cu circa de 20-30% cheltuielile materiale şi temporale la etapa de implementare şi a menţine toleranţa la defectări şi stabilitatea sistemului, având posibilitatea de a estima caracteristicile numerice de performanţă ale sistemului *OSR*.**

***Index Terms* — analiză, modelare, procese, reconfigurabilitate, sisteme orientate servicii.**

I. INTRODUCCERE

Orientarea pe servicii reprezintă o paradigmă *relevantă* pentru proiectarea şi implementarea modulară, explicită a aplicaţiilor de calcul distribuite concurente [1]. Acestea permit adaptarea la noi cerinţe sau modificări on-line de restricţii la procesarea aplicaţiilor de servicii reconfigurabile în timp real. Astfel, este posibil de a obţine interoperabilităţi şi interacţiuni dinamice între aplicaţii, indiferent de platformele şi limbajele de programare utilizate în dezvoltarea acestora.

Sistemele de calcul cu arhitecturi orientate pe servicii (SOA - Service Oriented Architecture) au o structură ierarhică cu mai multe nivele reconfigurabile, adaptându-se la schimbarea cerinţelor şi a mediului ambiant, restructurându-şi funcţionalitatea şi propriile configuraţii de hardware şi/sau software pe parcursul procesării aplicaţiilor prin adăugarea şi/sau înlăturarea unor componente sau resurse în sistem fără întreruperea procesării aplicaţiei curente [8].

Modelele propuse pentru descrierea adecvată a a acestor tipuri de sisteme devin mult mai complicate, iar cercetarea lor implică noi abordări [1].

Interesul nostru s-a îndreptat înspre reţele Petri reconfigurabile stocastice (*RPRS*) [3] ca model de referinţă, deoarece acestea constituie un formalism grafic simplu şi intuitiv de reprezentare a sistemelor cu evenimente discrete în care au loc fenomene de paralelism, de sincronizare, partajare a resurselor şi restructurare în timp real.

În lucrare sunt considerate unele aspecte de modelare şi analiză a proceselor de calcul orientate pe servicii reconfigurabile.

II. SISTEME DE CALCUL ORIENTATE PE SERVICII

Orientarea pe servicii reprezintă o nouă paradigmă de proiectare a sistemelor distribuite concurente [8]. Conceptele principale folosite în cadrul acestei abordări sunt:

• *serviciu*: unitate independentă a unei aplicaţii, care expune funcţionalitatea către clienţii din reţea prin intermediul unei interfeţe bazate pe mesaje;

• *client*: un modul al unei aplicaţii, care facilitează accesul utilizatorilor la funcţionalitatea oferită de servicii;

• *sistem distribuit*: un sistem interconectat de servicii şi clienţi.

Spre deosebire de sistemele distribuite orientate obiect, unde integrarea aplicaţiilor se face prin intermediul activării la distanţă a obiectelor, sistemele bazate pe servicii permit o integrare bazată pe schimbul de mesaje independente de modul de implementare al clientului şi al serviciului. Apelurile de metode din modelele distribuite orientate obiect sunt considerate în modelul orientat pe servicii o tehnică privată de implementare şi nu o construcţie de bază. Faptul că o interacţiune poate fi implementată ca un apel de metodă este considerat ca un detaliu de implementare, care nu este vizibil în exterior.

Sistemele orientate obiect sunt strâns cuplate; modificarea unei părţi poate fi realizată doar în condiţiile în care sunt modificate şi celelalte subsisteme, iar modificările sunt operate simultan. O asemenea abordare s-a dovedit a fi ineficientă, în special în situaţia în care diferitele subsisteme sunt realizate şi operate de către organizaţii diferite, cum este cazul aplicaţiilor de tip B2B (business-to-business). O altă problemă a acestor sisteme este cerinţa ca diferitele părţi ale sistemului trebuie să fie realizate folosind acelaşi model de obiecte şi aceeaşi platformă de dezvoltare.

Sistemele orientate pe servicii sunt sisteme slab cuplate, proiectate pentru a permite schimbarea pentru adaptarea la noi cerinţe sau metode de implementare. Prin utilizarea modelului orientat pe servicii este posibilă obţinerea unei interoperabilităţi între aplicaţii, indiferent de platformele şi limbajele de programare utilizate în dezvoltarea acesteia.

Dezvoltatorii şi integratorii de aplicaţii nu au nevoie să cunoască tipul de sistem, modelul de obiecte sau protocoalele folosite de către sistemele care trebuie integrate. Expunerea funcţionalităţii folosind protocoale standardizate şi interfeţe care pot fi obţinute dinamic conduce la o cuplare slabă între subsisteme, ceea ce permite o conectare simplă şi asigură o flexibilitate sporită aplicaţiei.

Dezvoltarea sistemelor orientate pe servicii se bazează pe patru principii fundamentale.

Frontierele între subsisteme sunt explicite: funcţionalitatea disponibilă în exterior este definită explicit la frontiera aplicaţiei (application boundary). Spre deosebire de metodele bazate pe invocarea implicită a metodelor, se foloseşte un model care impune construirea şi trimiterea explicită a mesajelor.

Deoarece sistemele orientate pe servicii sunt în general distribuite geografic şi rulează în medii sau organizaţii multiple, costul traversării frontielor este foarte mare din punctul de vedere al performanţelor şi al complexităţii. Arhitecturile orientate pe servicii iau în considerare acest lucru punând un accent deosebit pe identificarea şi definirea graniţelor dintre sisteme.

Serviciile care compun sistemul pot fi modificate dinamic şi puse în funcţiune independent faţă de restul sistemului. Această autonomie implică şi faptul că fiecare serviciu este complet responsabil de integritatea datelor şi proceselor pe care le controlează, având implicaţii şi asupra modului în care sunt tratate erorile. Pentru a menţine integritatea sistemului se pot folosi o serie de metode pentru tratarea unor astfel de cazuri: tranzacţii, sisteme robuste de mesagerie sau algoritmi special adaptaţi. Datorită faptului că multe servicii sunt disponibile prin reţele publice, sistemele trebuie să ia în considerare atât posibilitatea existenţei unor mesaje invalide, cât şi posibilitatea apariţiei unor mesaje trimise în scopuri maliţioase. Pentru protejarea sistemelor în astfel de cazuri şi pentru a permite scalare mai bună a sistemului, mesajele trimise trebuie să conţină în afară de datele efective şi informaţiile referitoare la contextul de execuţie şi dovezile necesare autorizării cererii.

Comunicarea între servicii este descrisă prin intermediul schemelor şi al contractelor şi nu prin clase: informaţiile necesare unui client pentru a folosi un serviciu sunt disponibile prin intermediul contractului care descrie structura mesajelor pentru cererile şi răspunsurile valide. Sistemele orientate obiect folosesc ca abstracţie de bază clasele pentru a grupa într-o singură entitate structura şi comportamentul aferent. Această grupare oferă avantaje la construirea modulelor aplicaţiei dar conduce la apariţia unor probleme legate de portabilitate între platforme şi necesită mecanisme mai complexe pentru comunicare. Arhitecturile bazate pe servicii propun o separare strictă între structură, descrisă prin schema mesajelor, şi comportamentul care este implementat privat în cadrul serviciului/clientului. Această separare permite o implementare mai uşoară a sistemelor prin renunţarea la cerinţele specifice sistemelor orientate obiect referitoare la compatibilitatea mediului de execuţie, contextul de securitate comun sau execuţia de cod transferat de la distanţă.

Compatibilitatea între servicii este bazată pe contracte. Pentru a permite obţinerea unei cuplări slabe între servicii, acestea trebuie să-şi publice cerinţele şi capabilităţile prin intermediul unor expresii prelucrabile automat. Aceste expresii definesc două tipuri de aserţiuni: condiţiile în care poate fi solicitat serviciul, cerinţe legate de forma şi conţinutul mesajului, garanţiile de securitate solicitate, şi garanţiile oferite de acesta. Aserţiunile sunt standardizate pentru a se asigura interpretare uniformă, indiferent de serviciul asupra căruia se aplică. Aceste aserţiuni oferă clienţilor o modalitate de a identifica serviciile compatibile şi permit serverului să efectueze o validare automată a cererilor înainte ca acestea să fie prelucrate efectiv.

La nivel global, structurarea arhitecturii SOA se poate face în trei straturi (Fig. 1): prezentare, aplicaţie şi acces la resurse.

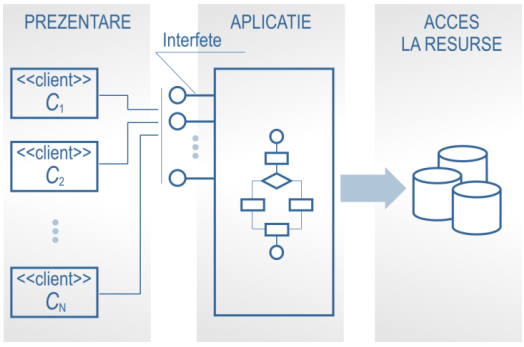

Fig. 1. Structurarea arhitecturii SOA.

Stratul de prezentare este compus din clienţii sistemului care facilitează accesul utilizatorilor la serviciile oferite de sistem. Aceste aplicaţii pot avea diferite forme, aplicaţii web, aplicaţii desktop, aplicaţii mobile, în funcţie de necesităţile clienţilor. Indiferent de tipul clientului, operaţiunile sunt efectuate unitar prin intermediul serviciilor sistemului.

Stratul de acces la resurse încapsulează resursele folosite de servicii pentru îndeplinirea operaţiilor. Aceste resurse pot fi surse de date, sisteme existente sau alte servicii.

Prelucrările care nu ţin de funcţionalitatea de bază a aplicaţiei (configurare, securitatea, instrumentarea, monitorizarea, auditul, validarea cererilor) pot fi separate sub forma aspectelor. Implementarea acestora este realizată folosind tehnica de interceptare a mesajelor care circulă între module. Această abordare permite simplificarea codului aplicaţiei şi permite dezvoltarea separată a elementelor care ţin de cerinţele secundare ale aplicaţiei.

Nucleul sistemului este constituit din stratul aplicaţiei. Acesta are două roluri principale: implementarea componentelor care asigură execuţia operaţiilor folosind resursele disponibile şi expunerea funcţionalităţii aplicaţiei către clienţi sub forma unui set stabil şi consistent de interfeţe. Separarea între interfeţele stabile şi componentele care le implementează permit sistemului să evolueze fără a necesita modificări în cadrul clienţilor sau a celorlalte sisteme care folosesc serviciile oferite.

Obiectivele generale urmărite în proiectarea arhitecturii sunt:

- asigurarea independenţei interfeţelor stabile de componentele care le implementează şi de mecanismele de transport ale mesajelor;
- să ofere un mecanism configurabil pentru rutarea cererilor primite către componentele care execută operaţiile şi a răspunsurilor primite de la acestea folosind mai multe canale de transport;
- să permită inserarea configurabilă în lanţul de procesare al mesajelor a codului corespunzător aspectelor.

În afară de obiectivele generale mai există şi o serie de cerinţe care trebuie îndeplinite de către serviciile individuale pentru a se asigura calitatea sistemului. Pentru ca serviciile să fie utile, durabile şi să ofere o performanţă satisfăcătoare în condiţiile unui număr mare de utilizatori

trebuie să îndeplinească funcţii utile întreprinderii, să aibă o granularitate corespunzătoare, să fie atomice şi să nu menţină informaţii de stare.

Serviciile trebuie de asemenea să nu menţină informaţii de stare între apelurile clienţilor. Toate informaţiile necesare sunt incluse în cadrul mesajului; contextul circulă împreună cu cererea şi răspunsul. Această abordare permite o distribuire eficientă a cererilor pe mai multe noduri de prelucrare şi permite implementarea separată a serviciilor secundare prin intermediul aspectelor.

Interacţiunea dintre aplicaţia client şi acţiunea executată de către sistem necesită rezolvarea mai multor probleme: specificarea modului de transfer al mesajelor, autorizarea cererilor, înregistrarea şi tratarea erorilor şi auditul acţiunilor întreprinse. Un exemplu de arhitectură ce ar permite rezolvarea acestor probleme şi ar facilita o comunicare eficientă şi flexibilă între servicii este ilustrat mai jos.

În afară de obiectivele generale mai există şi o serie de cerinţe care trebuie îndeplinite de către serviciile individuale pentru a se asigura calitatea sistemului. Pentru ca serviciile să fie utile, durabile şi să ofere o performanţă satisfăcătoare în condiţiile unui număr mare de utilizatori trebuie să îndeplinească funcţii utile întreprinderii, să aibă o granularitate corespunzătoare, să fie atomice şi să nu menţină informaţii de stare.

Serviciile trebuie de asemenea să nu menţină informaţii de stare între apelurile clienţilor. Toate informaţiile necesare sunt incluse în cadrul mesajului; contextul circulă împreună cu cererea şi răspunsul. Această abordare permite o distribuire eficientă a cererilor pe mai multe noduri de prelucrare şi permite implementarea separată a serviciilor secundare prin intermediul aspectelor.

Interacţiunea dintre aplicaţia client şi acţiunea executată de către sistem necesită rezolvarea mai multor probleme: specificarea modului de transfer al mesajelor, autorizarea cererilor, înregistrarea şi tratarea erorilor şi auditul acţiunilor întreprinse. Un exemplu de arhitectură ce ar permite rezolvarea acestor probleme şi ar facilita o comunicare eficientă şi flexibilă între servicii este ilustrat în Fig. 2.

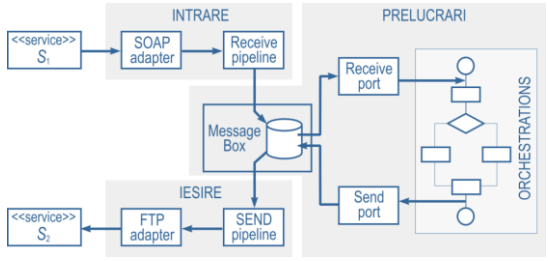

Fig. 2. Arhitectură de transfer şi prelucrare a mesajelor.

Datorită principiilor care stau la baza arhitecturilor orientate pe servicii reconfigurabile, modelarea şi analiza unor astfel de sisteme devine pe cât de actuală, pe atât şi de posibilă. Aceasta se datorează în mare parte decuplării consumatorului de implementarea serviciilor, precum şi abstractizării implementării serviciilor.

Daca privim serviciile ca niște cutii negre, fără a ține cont de specificul lor de implementare, atunci le putem descrie cu proprietăți formale cum ar fi capacitatea de prelucrare, latența de prelucrare, tipurile de mesaje excitante etc.

La fel, trecerile de la o etapă la alta în cadrul procesului de lucru pot fi caracterizate cu proprietățile lor specifice. La etapa inițială de proiectare datele privind valorile concrete ale unor proprietăți pot fi luate din cerințe sau din rezultatele testării fiecărui modul în parte.

Modelarea funcționării sistemelor construite pe principii SOA reduce riscul implementării unui sistem ce nu corespunde cerințelor operaționale. Pe parcursul modelării se obțin rezultate, care fiind ulterior analizate pot sugera configurații optime de servicii.

Instrumentul cel mai potrivit pentru modelarea sistemelor bazate pe arhitecturi orientate pe servicii sunt rețelele Petri. Fiind extinse, rețelele Petri oferă o acoperire tot mai largă a problemelor legate de evaluarea performanțelor sistemelor cu calcul distribuit şi paralel. Pentru arhitecturile ce presupun înregistrarea dinamică a serviciilor şi consumatorilor sunt binevenite rețelele Petri descriptiv reconfigurabile stocastice.

III. REŢELE PETRI RECONFIGURABILE

Reţele Petri reconfigurabile stocastice (*RPRS*), care reprezintă o extensie a reţelelor Petri generalizate stocastice (*RPGS*) [2, 4, 6], prezintă un set de calităţi foarte importante şi anume:

- dispun de o definiţie formală, caracterul căreia permite de a produce specificaţii lipsite de ambiguităţi, încât fiecare model astfel construit posedă o semantică perfect definită;
- au o mare putere de expresie, deoarece anume sunt foarte bine adaptate pentru a descrie comportamente complexe reactive sau concurente;
- sunt executable: modelele de *RPRS* pot fi interpretate de către un program construit dintr-o definiţie formală de notaţii, ceia ce permite de a simula funcţionarea sistemului pe parcursul specificaţiei. Conceptorul profită astfel de o viziume dinamică a sistemului pentru a înţelge mai profund comportamentul său;
- sunt disponibile tehnici de verificare automată a proprietăţilor modelului. Este posibil de a determina proprietăţi generice de mărginire, viabilitate, reversibilitate, accesibilitate etc. sau cele specitfice de existenţă a invarianţilor;
- au o reprezentare grafică atrăgătoare, care măreşte lizibilitatea şi facilitează înţălegerea modelelor. Aceata este foarte utilă la derularea interactivă vizuală a modelelor pentru ajustarea lor.

Pentru a trata unele probleme menţionate la modelarea şi evaluarea proceselor de calcul reconfigurabil, în este un model de reţele *RPGS dinamic descriptiv-restructurabile*, în care introducem o mulţime de reguli de rescriere $R = \{r_1, ..., r_k\}$ a reţelei *RDM* curente ce poate modifica atât marcajul, cât şi structura ei la ocurenţa unor evenimente specificate. O regulă de rescriere $r_j \in R$ este o generalizare a noţiunii de tranziţie discretă $t_j \in T_d$, folosită în sens clasic. Condiţia de validare de către marcajul curent M a unei tranziţii $t_j \in T_d$ şi/sau reguli $r_j \in R$ sunt similare.

Pentru a facilita redarea lucrării, prezentăm succint definirea *RPRS*: $RN =< \Gamma, R, \phi, G_{tr}\ G_r, M, \Lambda >$, unde Γ = < *P, T, Pre, Post, Test, Inh,* K_p*, G, Pri* > este o reţea *RPGS*; $R = \{r_1, ..., r_k\}$ – o mulţime finită de reguli de

reconfigurare; $\phi: E \to \{T, R\}$, $E = T \cup R$ – funcţie, care indică tipul evenimentului validat ce poate fi declanşat; G_r, $G_{tr}: R \times Bag(P) \to \{true, false\}$ respectiv sunt *funcţii de gardă* asociate cu fiecare regulă *r* de reconfigurare a reţelei. Implicit $\forall r \in R$ în marcajul curent $M: g_{tr}(M) \in G_{tr}$ este "*true*", iar $g_r(M) \in G_r$ este "*false*". Aceste funcţii sunt booleane şi se calculează la fiecare iteraţie de marcare a reţelei, iar in cazul îndeplinirii condiţiei ele modifică reteaua iniţială. Validarea unei reguli *r* în *RPRS* este efectuată cum şi pentru *RPG* când $g_{tr}(M)$ este "*true*". La declanşarea lui *r*, în cazul în care $g_r(M)$ are valoarea "*false*", *r* va schimba numai marcajul curent M al *RPRS* într-un marcaj nou M'. Însă dacă $g_r(M)$ are valoarea "*true*", *r* va reconfigura reţeaua RN în altă reţea RN' după regula: $r: RN_L \triangleright RN_W$.

Modificarea retelei RN are loc în două etape. La prima etapă are loc eliminarea elementelor specificate de sub-reţeaua RN_L ale reţelei curente, obţinând o nouă reţea $RN\rho = RN \setminus RN_L$. Etapa a doua constă din adaugarea sub-reţelei RN_W în reţeaua $RN\rho$ curente cu elemente noi ale reţelei, obţinând astfel $RN' = RN\rho \cup RN_W$. Dacă în reţeaua curentă sunt elemente cu acelaş nume ca la elementele ce urmează a fi adaugate, ele se suprascriu, sau cu alte cuvinte se contopesc. Astfel, ca rezultat se va obţine o reţea nouă RN', care va funcţiona după noi reguli. La o definire corespunzatoare a funcţiilor de reconfigurare este posibilă funcţionarea reţelei în regim flip-flop, adică în unele condiţii reţeaua se modifică intr-un anumit fel, iar apoi poate reveni la configurarea iniţială.

Menţionăm, că o tranziţie t_{r_j} este un caz particular al unei reguli de rescriere r_j a reţelei *RN*, deoarece, pentru $g_r(M) =$ "*false*", declanşarea ei va rescrie numai marcajul curent în alt marcaj nou al reţelei, însă pentru $g_r(M) =$"*true*", declanşarea ei va rescrie atât structura reţelei curente, cât şi marcajul curent al reţelei într-o altă sructură cu un marcaj nou.

IV. ASPECTE DE ANALIZĂ A SERVICIILOR RECONFIGURABILE

Compunerea unui *model* de reţea *RPRS*, ce descrie funcţionarea unui *sistem de calcul orientat pe servicii reconfigurabile* sau în curs de realizare, este deci *un aspect crucial şi mult mai dificil* decât verificarea şi analiza lui (printr-o metodă exactă sau aproximativă) şi interpretarea rezultatelor obţinute. Cea mai mare dificultate rezidă în faptul că procesele din cadrul unui astfel de sistem folosesc, la un moment dat, mai mult de o resursă de calcul. Altă dificultate este generată de posibilitatea ca o resursă de calcul să fie privită (în dependenţă de context) ca o staţie activă sau ca o resursă pasivă, starea căreea se schimbă in mod dinamic. În plus, deoarece viteza de funcţionare a diverselor elemente de prelucrare şi a subsistemelor de comunicaţie diferă prin mai multe ordine de mărime, manipularea unui model, care încorporează toate resursele sistemului, este deosebit de dificilă. O abordare practică pentru a conturna astfel de dificultăţi constă în structurarea modelului de sistem, luând în consideraţie abordarea complexităţii sistemului prin partiţionarea sa în subsisteme cooperante.

Pentru a aborda aceste dificultăţi este necesar:

1. De a efectua analiza cerinţelor către sistemul elaborat, unde sunt determinaţi parametrii externi, care descriu sistemul din punct de vedere al bineficiarilor, iar apoi sunt specificate restricţiile de funcţionare;
2. A determina criteriile de formalizare ce vor permite de a estima deciziile în procesul de proiectare, apoi elabora modelul sistemului la diferite niveluri de abstractizare: la nivel static, nivel de procese şi nivel de servicii.
3. La nivel static este descrisă arhitectura sistemului orientat pe sevicii. La nivel de de procese sunt descrise interconexiunile, interdependeţele între servicii şi dinamica interacţiunilor, modelul funcţionării cărora este redat de reţele Petri reconfigurabile stocastice (*RPRS*) [3].

Astfel, modelarea proceselor sistemelor de calcul *orientate pe servicii* reconfigurabile se va efectua prin reţele *RPRS* la niveluri de stări. Pentru aceasta se va determina mul'imea de acţiuni ce se produc în sistem, care stări preced acestor acţiuni şi în ce stări va trece sistemul după producerea acestor acţiuni.

Folosind abordarea de construire a modelelor de reţele *RPRS* descriptiv-compoziţionale, ce exprimă cooperarea proceselor de calcul [3, 4] ce descrie funcţionarea unui sistem cu evenimente discrete, modelarea acestora constă în a exprima, cu un anumit nivel de detaliere, comportamentul logic al interacţiunii proceselor cooperante ale (sub)sistemului considerat. Având o descriere informală, aceasta implică faptul că la elaborarea modelului de reţea *RPRS* este indispensabil de a cunoaşte:

– structura, componentele sistemului, atributele şi stările lor locale, care pot să le primească;
– condiţiile şi evenimentele ce pot schimba stările, fie că ele provin de la un proces aleatoriu, fie de la o decizie specificată;
– condiţiile de sincronizare şi cooperare, care determină ocurenţa unor evenimente, etc.;
– atributele calitative şi cantitative ce determină restricţiile de funcţionare ale sistemului, etc;
– specificaţiile condiţiilor interacţiunii evenimentelor şi nivelul de detaliere a modelării, redate de o descriere informală a funcţionării sistemului considerat.

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

La nivelul modelului static sunt descrise arhitectura, atributele şi structura ale componentelor sistemului orientat pe servicii.

Fluxurile de mesaje şi lucrări sunt descrise prin reţele *RPRS*. Fiecare serviciu este prezentat în formă de modul aparte cu intrările şi ieşirile sale. Fiecare serviciu este caracterizat de un set de operaţii (metode). Interfaţa serviciului este reprezentată de un set de locaţii de intrare. Jetoanele modelului în locaţiile interfeţelor sunt interpretate ca lucrări (mesaje) de prelucrare a informaţiei (redate de tranziţii temporizate) transmise serviciului pe canale asincrone legătură date. Ordinea recepţiei mesajelor poate să difere de cea de emitere de către serviciul sursă.

Locaţiile de ieşire ale modulului serviciu reprezintă transmiterea fluxului de control altor servicii pentru prelucrarea altor informaţii.

Un modul srviciu S_i cu atributele sale este descris de o subreţea $RPRS_i$.

Modelul nivelului de procese servicii descrie interconexiunile şi interacţiunile între diferite servicii.

Pentru a efectua compunerea modelelor sistemelor de calcul în formă de expresii descriptive ale *RPRS* [3, 4], vom efectua următoarele etape:

1. Se va determina nivelul de detaliere al sistemului şi respectiv al modelului. Pentru aceasta se vor evidenţia aparte acele componente ale sistemului, funcţionarea cărora poate fi reprezentată în forma de interacţiunea unor procese cooperante, de sincronizare, replicare, etc.;

2. Pentru fiecare tip de elemente, componente se va descrie traseul procesului ce determină fazele de funcţionare, stările locale respective, în care poate să se afle fiecare element şi logica schimbării acestor stări;

3. Se vor identifica starile locale ale procesului, specificând astfel variabilele descriptive ce determină semantica elementelor, resursele alocate, relaţiile temporale ale evenimentelor, pentru a accesa aceste stări;

4. Se vor determina atributele numerice pentru elementele de fiecare tip şi starea lor iniţială;

5. Pentru fiecare traseu al procesului ce determină schimbul de faze (stări locale) ale sub-sistemelor cooperante, folosind operatorii necesari, se va compune o expresie descriptivă care redă o subreţea *RPRS* elementară, unde fiecărei faze (stări locale) a procesului îi va corespunde o locaţie, şi schimbului de faze îi va corespunde o tranziţie specificată, iar numărul de elemente resurse în aceste faze corespunde numărului de jetoane în locaţie cu capacitatea respectivă;

6. Se va determina semantica şi atributele numerice ale locaţiilor şi tranziţiilor subreţelelor *RPRS*, luând în consideraţie politicile de executare a tranziţiilor validate (declanşarea prioritară, funcţiile de gardă, selectarea probabilistică, etc., dacă ele sunt folosite în sistem);

7. Folosind operatorii respectivi de compunere a submodelelor şi operaţia de contopire a locaţiilor şi tranziţiilor etichetate ce au aceeaşi esenţă semantică ale subreţelelor *RPRS*, redate de expresiile descriptive identificate, obţinem modelul global al sistemului în formă de reţea *RPRS* plată.

Reţeaua *RPRS* astfel construită, urmând metoda redată mai sus, descrie într-o formă explicită cooperarea, paralelismul, sincronizarea, concurenţa, excluderea mutuală, reconfigurabilitatea şi alte forme de interacţiuni ale proceselor concurente ale sistemului modelat.

Aceasta presupune efectuarea următoarelor sarcini:

1. Estimarea restricţiilor comportamentale temporale şi a cerinţelor de performanţă faţă de folosirea resurselor de calcul;

2. Verificarea corectitudinii interacţiunii serviciilor efectuate. Aceasta presupune analiza proprietăţilor calitative ale modelului: viabilitate, mărginire, lipsa de blocage etc.;

3. Balansarea sistemului efectuată prin determinarea locurilor înguste (componente unde apar congestii) şi repartizarea uniformă a încărcării coponentelor sistemului;

4. Estimarea toleranţei la defectări luând în consideraţie efectele disfuncţiunii stocastice ale unor componente de servcii.

Pentru efectua analiza proprietăţilor comportamentale dinamice şi structurale ale modelelor de reţele *RPRS* astfel construite vom folosi metode şi tehnici descrise în [4], aplicând un mediu instrumental [7], de exemplu, Sistememul Software de Simulare Animată a Reţelelor Petri Diferenţiale Reconfigurabile. elaborat şi realizat de autori [9].

V. CONCLUSIE

Analiza proprietăţilor comportamantale şi verificarea modelului la etapa de proiectare a sistemelor orientate pe servicii reconfigurabile, folosind abordarea considerată în această lucrare permite: de a diminua cu circa de 20-30% cheltuielile materiale şi temporale la etapa de implementare; a menţine toleranţa la defectări şi stabilitatea sistemului, având posibilitatea de a estima caracteristicile numerice de performanţă ale sistemului.

Acest tip de modele permite de a descrie în mod natural procesele de calcul reconfigurabile.

BIBLIOGRAFIE

[1] M. Bell, Service-Oriented Modeling (SOA): Service Analysis, Design, and Architecture, Wiley, 2008.

[2] E. Guţuleac, Evaluarea performanţelor proceselor de calcul prin reţele Petri generalizate stocastice. Meridian ingineresc, Nr. 1, 2008, Ed.: UTM, Chişinău, p.27-32.

[3] E. Guţuleac, M. L. Mocanu, Iu. Ţurcanu, „Dynamic Rewriting of Differential Petri Nets for Modeling of Hybrid Systems", Proceedings of the 2nd International Conference on Intelligent Computer Communication and Processing (ICCP 2006), Cluj-Napoca, 1-2 Sept., România, 2006, p. 105-112,

[4] E. Guţuleac, Evaluarea performanţelor sistemelor de calcul prin reţele Petri stochastice, Ed.: „Tehnica-Info", Chişinău, 2004, - 276 p.

[5] K. Compton, S. Hauck, "Reconfigurable Computing: a Survey of Systems and Software," ACM Computing Surveys (CSUR), vol. 34, no. 2, pp. 171-210, 1998.

[6] M. Llorens, J. Oliver, "Structural and Dynamic Changes in Concurrent Systems: Reconfigurable Nets," IEEE Transactions on Computers, vol. 53, no. 9, pp. 1147-1158, 2004.

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

[8] M. Rosen, B. Lublinsky, K. T. Smith, M, J. Balcer, Applied SOA: Service-Oriented Architecture and Design Strategies, Wiley, 2008.

[9] Iu. Ţurcanu, E. Guţuleac, A. Cordunenu. Sistem Software de Simulare Animată a Reţelelor Petri Diferenţiale Reconfigurabile. Proceedings of the 6-th International Conference on Microelectronics and Computer Science, ICMCS 2009, vol. 1, 1-3 October, 2009, p. 307-311.