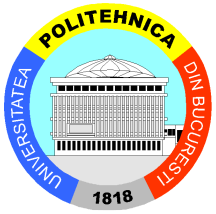

UNIVERSITATEA **POLITEHNICA** DIN BUCUREŞTI
Facultatea Automatică şi Calculatoare
Departamentul de Calculatoare

Compunerea serviciilor web bazată pe calitatea serviciilor

Rezumat

Autor: Ing. Raluca Iordache

Conducător ştiinţific:
Prof. Dr. Ing. Florica Moldoveanu

Bucureşti 2014

Cuprins

1. Introducere

1.1 Motivaţie. Obiectivul tezei.

În momentul actual, arhitecturile bazate pe servicii oferă oportunitatea dezvoltării de noi aplicaţii complexe prin compunerea unor aplicaţii deja existente, oferite ca servicii, pentru a reacţiona cerinţelor pieţei de a crea rapid aplicaţii noi. Arhitecturile bazate pe servicii permit selecţia dinamică şi integrarea automată a serviciilor componente oferind soluţii flexibile şi adaptabile la problemele care pot apărea în timpul execuţiei.

Un serviciu compus este descris de un model abstract de compunere care combină o serie de activităţi. Pentru a putea executa un serviciu compus, fiecărei activităţi trebuie să îi fie asociat un serviciu web concret care îndeplineşte funcţionalitatea cerută.

Existenţa unei oferte mari de servicii web ce implementează funcţionalităţi similare, sau chiar identice, face necesară investigarea unor posibilităţi de comparare a serviciilor web pe baza unor criterii legate de calitatea serviciilor (QoS). Compararea serviciilor trebuie sa ţină cont de cerinţele clientului, iar găsirea serviciului web optim pentru cerinţele date este o sarcină complexă. În practică există în mod curent situaţii în care nici o soluţie nu indeplineşte toate criteriile clientului, deoarece trebuie urmărite mai multe cerinţe, de obicei conflictuale. Un exemplu ar fi cazul des întâlnit în care se doreşte un serviciu web care să ofere un timp de răspuns mic, indisponibilitate a sistemului redusă şi în acelaşi timp un preţ mic. În această situaţie este posibil să nu existe o soluţie care să optimizeze toate criteriile, aşa că în acest caz soluţia cea mai bună este soluţia care conţine cel mai bun compromis. Clientul trebuie sa aibă posibilitatea de a specifica regulile pentru alegerea unei soluţii de compromis optime.

În cazul serviciilor web compuse, trebuie calculată valoarea agregată a calităţii serviciilor pentru modelul de orchestrare care defineşte compoziţia respectivă.

Serviciile web sunt componente dinamice: în orice moment un serviciu poate dispărea şi noi servicii similare pot apărea. Din acest motiv, caracteristicile nefuncţionale QoS se pot modifica frecvent. Astfel, se impune o abordare automată şi dinamică pentru compunerea serviciilor web.

În prezent există o serie de abordări care adresează problema compunerii serviciilor web ţinând cont de optimizarea parametrilor QoS, dar nu există nici o soluţie standard în această direcţie de cercetare, care să ţină cont atât de cerinţele cât şi de preferinţele clientului.

Obiectivul acestei teze de doctorat este de a furniza o metodă eficientă, dar în acelaşi timp intuitivă şi simplu de utilizat, de compunere a serviciilor web bazată pe cerinţele şi preferinţele clienţilor. Pentru specificarea preferinţelor clienţilor ne-am propus să concepem un limbaj expresiv, care să permită formularea de reguli complexe. Abordarea trebuie să fie completă, pornind de la căutarea semantică a serviciilor şi continuând cu selecţia serviciilor optime pentru activităţile modelului de compunere. Optimizarea compunerii trebuie să ţină cont de preferinţele clientului, referitoare la parametrii de calitate ai serviciului compus rezultat.

1.2 Contribuţii personale

Teza de doctorat propune o abordare completă pentru compunerea serviciilor web bazată pe calitatea serviciilor. Această abordare distinge două etape importante: descoperirea serviciilor web candidat şi alegerea serviciilor optime ţinând cont de cerinţele de calitate ale clientului. Contribuţiile abordării propuse sunt atât teoretice cât şi practice, fiind oferite ca proiecte open-source.

Descoperirea serviciilor pe baza unei căutări sintactice oferă rezultate cu o acurateţe scăzută. Această problemă este rezolvată folosind în etapa de descoperire a serviciilor o căutare semantică, bazată pe ontologii, pentru a identifica pentru fiecare activitate din modelul de compunere abstract o listă de servicii candidat care respectă atât cerinţele funcţionale cât şi cerinţele QoS. În etapa de selecţie a serviciilor, pentru fiecare activitate din modelul de orchestare este ales un serviciu web optim, din lista de servicii candidat oferită de etapa anterioară. Procesul de selecţie adoptă o strategie de optimizare globală pentru a realiza compunerea de servicii care respectă cel mai bine preferinţele QoS agregate specificate de client. O noutate a abordării noastre este integrarea preferinţelor QoS în cadrul procesului de descoperire a serviciilor, pentru a restrânge numărul de servicii candidat pentru fiecare activitate din modelul de orchestrare. În acest scop oferim o extensie a unei ontologii de preferinţe, care oferă posibilitatea specificării preferinţelor şi a regulilor de compromis şi folosim o strategie de optimizare locală în cadrul etapei de descoperire.

O altă contribuţie a tezei este propunerea unei metode de specificare a proprietăţilor QoS, denumită QoSPref [**1**], bazată pe un limbaj simplu şi intuitiv care oferă posibilitatea exprimării unor preferinţe complexe şi a unor reguli de compromis. În cadrul acestei metode, propunem un algoritm de selecţie a serviciilor web simplu, care nu necesită technici complexe de decizie multicriterială. Algoritmul nostru realizează o ordonare totală a alternativelor şi poate accepta şi preferinţe intranzitive.

Pentru a putea folosi şi testa metoda propusă, oferim o implementare a acestei metode, sub forma unui proiect open-source. Aplicaţia noastră oferă şi o interfaţă grafică pentru configurarea serviciilor candidat şi a caracteristicilor acestora, precum şi pentru configurarea cerinţelor şi preferinţelor clienţilor. Proiectul open-source este disponibil la adresa http://sourceforge.net/projects/qospref/.

Pentru a oferi o ontologie de preferinţe care să folosească flexibilitatea metodei noastre de specificare a preferinţelor (QoSPref) şi a regulilor de compromis, am extins ontologia de preferinţe existentă OWL-Q [**2**]. Această ontologie oferă posbilitatea specificării simetrice a cerinţelor clientului şi a caracteristicilor serviciilor oferite.

Folosind o căutare semantică bazată pe cerinţele funcţionale şi nefuncţionale ale clientului, obţinem o listă de servicii candidat pentru fiecare activitate din modelul abstract de orchestrare [**3**]. Pentru a alege pentru fiecare activitate serviciul concret care va îndeplini funcţionalitatea cerută trebuie să calculăm şi să comparăm valorile QoS agregate ale

serviciului compus. Calcularea valorii QoS agregate pentru un serviciu compus este o problemă complexă, iar majoritatea soluţiilor existente se limitează la modele de compunere care pot fi reprezentate ca workflowuri structurate. Soluţia oferită de noi pentru alegerea serviciilor concrete reprezintă o abordare *binding-as-a-service* (BaaS) [**4**] şi foloseşte metoda propusă de Yang et al. [**5**] pentru a elimina această restricţie.

Serviciile concrete care oferă soluţia optimă compusă sunt alese folosind un algoritm genetic [**6**] care evaluează fitness-ul unei soluţii pe baza valorii agregate QoS şi a preferinţelor specificate de client folosind metoda noastră QoSPref.

Abordarea noastră *binding-as-a-service* (BaaS) este oferită sub forma unui serviciu web care permite alegerea serviciilor concrete din cadrul unei mulţimi de serivicii candidat, pe baza modelului abstract de compunere şi având in vedere optimizarea globală a serviciului compus. BaaS este implementat sub forma unui proiect open-source, disponibil la adresa: http://sourceforge.net/projects/baas/.

Metoda QoSPref poate fi aplicată nu numai pentru selecţia serviciilor pe baza caracteristicilor QoS, ci şi în cadrul general al oricărei probleme de optimizare multiobiectiv. Pentru a analiza utilitatea unei astfel de abordări, am adaptat metoda QoSPref şi am integrat-o într-un framework pentru studiul tehnicilor de incorporare a preferinţelor în algoritmi evolutivi de rezolvare a problemelor multiobiectiv [**7**].

1.3 Lucrări stiinţifice în care au fost publicate contribuţiile originale cuprinse în teză

- Raluca Iordache and Florica Moldoveanu, "A conditional lexicographic approach for the elicitation of QoS Preferences," in *Lecture Notes in Computer Science vol. 7565, pp 182-193*, Rome, 2012 (rata de acceptare: 20%; indexat ISI).

- Raluca Iordache and Florica Moldoveanu, "QoS-aware web service semantic selection based on preferences," in *24th DAAAM International Symposium on Intelligent Manufacturing and Automation*, Zadar, Croatia, Procedia Engineering (2014), pp. 1152-1161.

- R. Iordache and F. Moldoveanu, "A web service composition approach based on QoS preferences," in *Proceedings of the 6th IEEE International Conference on Service Oriented Computing & Applications (SOCA 2013)*, Kauai, Hawaii, 2013, p. 220-224 (rata de acceptare: 27%).

- Raluca Iordache and Florica Moldoveanu, "A genetic algorithm for automated service binding," in *24th DAAAM International Symposium on Intelligent Manufacturing and Automation*, Zadar, Croatia, Procedia Engineering (2014), pp. 1162-1171.

- Raluca Iordache and Florica Moldoveanu, “An efficient approach for an end to end web service composition based on QoS preferences” în UPB Scientific Bulletin (în curs de apariţie).

- Raluca Iordache, Serban Iordache and Florica Moldoveanu, "A Framework for the Study of Preference Incorporation in Multiobjective Evolutionary Algorithms," in Proceedings of the 16th Annual Conference on Genetic and Evolutionary Computation, GECCO 2014, Vancouver, Canada, ACM (în curs de apariţie; indexat ISI).

1.4 Organizarea tezei

Teza de doctorat este formată din zece capitole. Primele şase capitole prezintă domeniul de studiu, stadiul actual al cercetării şi problemele deschise existente. Următoarele trei capitole reprezintă contribuţii originale, menite să rezolve unele dintre problemele menţionate în capitolele anterioare. Ultimul capitol prezintă concluziile activităţii de cercetare.

Capitolul 1 prezintă problemele existente în prezent în cadrul direcţiei de cercetare a compunerii serviciilor web bazate pe cerinţele şi preferinţele clientului şi formulează obiectivele tezei de doctorat.

Capitolul 2 prezintă noţiuni introductive şi stadiul actual al cercetării în domeniul serviciilor web şi al arhitecturilor bazate pe servicii.

Capitolul 3 prezintă stadiul actual al cercetării dedicate compunerii serviciilor web, standardele existente şi cele mai importante abordări propuse în această direcţie.

Capitolul 4 prezintă stadiul actual al cercetării în domeniul calităţii serviciilor web şi metodele propuse de adaptare a standardelor existente pentru a încorpora atributele de calitate QoS.

Capitolul 5 prezintă provocările aduse de compunerea serviciilor web bazată pe calitatea serviciilor şi prezintă metodele de calcul al calităţii pentru o compunere de servicii şi modalităţile de optimizare ale acesteia.

Selecţia bazată pe calitatea serviciilor QoS este o problemă de optimizare multiobiectiv, iar Capitolul 6 tratează metodele de rezolvare ale acestei probleme prezentând abordările tradiţionale precum şi metodele evolutive.

În Capitolul 7 propunem o metodă inovativă de specificare a preferinţelor denumită QoSPref,folosind un limbaj simplu şi intuitiv dar care permite în acelasi timp exprimarea unor preferinţe complexe. Aici este prezentată şi aplicaţia care implementează această metodă şi care demonstrează eficacitatea metodei propuse.

Capitolul 8 îşi îndreaptă atenţia asupra căutarii serviciilor web ţinând cont de atributele QoS şi prezintă contribuţia tezei în această direcţie. Abordarea propusăîn acest capitol este bazată pe extinderea ontologiei de preferinţe OWL-Q pentru a permite descrierea preferinţelor în limbajul QoSPref introdus în capitolul anterior.

Capitolul 9 tratează problema selecţiei serviciilor web în cazul modelelor complexe de compunere ţinând cont de calitatea globală a compunerii. Abordarea propusă combină algoritmul de selecţie QoSPref cu o metodă eficientă de calcul al valorii QoS agregate pe baza unor modele de orchestrare care pot conţine şi componente nestructurate. Implementarea pe care o oferim pentru metoda propusă foloseşte pentru alegerea soluţiei optime un algoritm genetic modificat.

Capitolul 10 prezintă concluziile activităţii de cercetare ale carei rezultate sunt prezentate referitoare la contribuţiile propuse şi direcţiile viitoare de cercetare.

2. Fundamente teoretice

2.1 SOA

SOA propune câteva principii arhitecturale pentru dezvoltarea și integrarea aplicațiilor software, neimpunând adoptarea unei anumite tehnologii pentru implementare. SOA se bazează în prezent pe serviciile web, acestea fiind acceptate în mod universal ca un standard în industrie, dar SOA poate fi în principiu implementată folosind orice tip de tehnologie bazată pe servicii. În cadrul arhitecturilor SOA, în cazul în care cerinţele unei aplicaţii nu pot fi îndeplinite de către un singur serviciu web, mai multe servicii şi procese existente pe Internet sunt interconectate pentru a îndeplini cerinţele clientului. Astfel, fiecare scenariu al unei aplicaţii este împărţit în sarcini ce pot fi realizate de către servicii atomice deja existente. Cu toate că în prezent există numeroase abordări atât practice cât şi teoretice în această direcţie, compunerea serviciilor web rămâne o problemă de actualitate complexă. Complexitatea este cauzată în principal de numărul mare de servicii existente ce trebuie analizate şi a eterogenităţii acestora. De asemenea, aplicaţia finală trebuie să respecte criteriile de calitate cerute de client.

2.2 Compunerea serviciilor Web

Existența unei oferte mari de servicii web aduce cu sine necesitatea dezvoltării unui mecanism de compunere a serviciilor pentru realizarea unui proces de afaceri complex ce îmbină funcționalitatea mai multor servicii existente, în scopul obținerii funcţionalității dorite. Rezultatul obținut se numește compunere a serviciilor, iar în momentul în care acest serviciu este oferit ca un serviciu de sine stătător, el se numește serviciu compus.

Compunerea serviciilor web ridică numeroase probleme legate de posibilitatea descoperirii serviciilor web necesare şi de asigurarea corectitudinii unei soluţii de compunere. Compunerea serviciilor se diferenţiază în funcție de gradul de automatizare în: compunere *manuală*, *asistată* şi *automată* a serviciilor. Compunerea serviciilor realizată în momentul proiectării implică o compunere statică, realizată în faza de dezvoltare a aplicaţiei sau sistemului. O compunere realizată în faza de rulare este o compunere dinamică, creată şi optimizată pentru cererea din momentul respectiv. Scenariile practice necesită o compunere automată în care o aplicație realizează compunerea serviciilor, fără intervenție umană. Pentru realizarea unei compuneri automate a serviciilor este nevoie ca serviciile să aibă o descriere semantică.

Compunerea serviciilor se realizează în următorii paşi:

- Identificarea scopului compunerii
- Selectarea serviciilor
- Ordonarea corectă a serviciilor (controlul intrărilor şi ieşirilor şi al datelor transmise)
- Execuţia compunerii

- Verificarea compunerii obţinute, monitorizarea soluţiei
- Adaptarea, modificarea, în funcţie de rezultatele verificării şi montorizării.

BPM (Business Process Management) este o una dintre abordările cele mai des folosite pentru gestionarea proceselor, având ca scop alinierea tuturor aspectelor unei organizații cu cerințele clienților. BPM are în vedere optimizarea activităților curente ale unei organizații și a interacțiunilor umane din interiorul și cu exteriorul firmei, operând cu acestea sub formă de procese. Un proces de afaceri reprezintă o serie de activități. Împreună, activitățile realizează un anumit scop. BPMS (Business Process Management Systems) coordonează procesele de afaceri provenind dintr-un mediu inter-organizaţional.

Pentru a putea gestiona procesele de afaceri, este nevoie ca acestea să poată fi descrise și documentate. Pentru descrierea proceselor complexe și a tuturor aspectelor relevante (cum ar fi fluxul de date, structurile de control, evenimentele) este nevoie de utilizarea unei notații.

Pentru modelarea proceselor de afaceri s-a evidențiat şi este folosit pe scară largă Business Process Management Notation (BPMN) [**8**]. Acest limbaj de modelare a proceselor pune accentul pe controlul fluxului și oferă un limbaj comun de modelare, ce poate fi folosit atât de către utilizatorul tehnic, cât şi de utilizatorul de afaceri, păstrând în acelaşi timp nealterată semantica procesului de afaceri. Cu alte cuvinte, BPMN creează o punte standardizată pentru lacuna dintre procesul de proiectare a afacerilor şi procesul de implementare. Acest standard pentru modelarea proceselor s-a impus pe scară largă, existând în prezent un număr mare de aplicaţii care oferă posibiltatea de modelare BPMN.

2.3 Calitatea serviciilor Web (QoS)

În momentul actual, existenţa unei oferte mari de servicii web ce implementează funcţionalităţi similare sau chiar identice face necesară investigarea unor posibilităţi de comparare a serviciilor web pe baza unor criterii legate de calitatea serviciilor (QoS).

QoS este definit în ISO 8402 [**9**] ca “totalitatea atributelor și caracteristicilor unui produs sau serviciu, care influențează capacitatea acestuia de a satisface anumite cereri stabilite sau implicite”. QoS este un concept larg, ce cuprinde atribute importante, funcţionale şi nefuncţionale, cum ar fi: metrici de performanţă, atribute de securitate, integritate tranzacţională, scalabilitate, fiabilitate, disponibilitate - caracteristici ce sunt definite pentru fiecare serviciu în parte.

Un atribut de calitate nu este însă o măsură cantitativă. Spre exemplu, pentru specificarea duratei de execuție a unui serviciu nu se poate indica un număr exact de milisecunde, ci se pot defini alte sub-atribute care să descrie acest atribut (spre exemplu: durata maximă de răspuns, durata medie de execuție). Aceste sub-atribute sunt cunoscute sub numele de *metrici de calitate.* În [**10**] o metrică de calitate este definită ca “o măsură cantitativă a gradului în care un element deține un atribut de calitate”. Astfel, atributul de calitate “disponibilitate” poate fi definit folosind următoarele metrici: “disponibilitate medie”,

reprezentând timpul în care un serviciu este accesibil, definit in procente, şi “interval între erori ”, reprezentând intervalul de timp mediu dintre două eșuări ale serviciului dat [**11**]. Calitatea unui serviciu compus din alte servicii web este dată de calitatea fiecărui serviciu implicat în compunere, în parte. QoS reprezintă un factor important pentru asigurarea satisfacţiei clientului, însă atributele de calitate catalogate drept importante de un anumit client sunt specifice nevoilor sale şi sunt diferite de cele ale altor clienţi. Spre exemplu, pentru un client, atribute cum ar fi durata execuţiei şi securitatea pot fi cruciale, în timp ce pentru un alt client aspectele legate de preţ sunt mai importante decât durata execuţiei. Astfel, se impune necesitatea realizării unei metode de compunere a serviciilor web ţinând cont de calitatea serviciilor cerută de un anumit client.

2.4 Compunerea serviciilor web în funcţie de parametrii QoS

2.4.1 Optimizarea compunerii serviciilor web in funcție de QoS

În cadrul compunerii de servicii se remarcă sistemele de tip workflow (Workflow Management Systems). În prezent există sisteme de tip workflow dinamice şi flexibile ce oferă posibilitatea selecţiei atomate a serviciilor concrete ce pot implementa serviciile abstracte ce definesc un proces, din punct de vedere al funcţionalităţii oferite. Această funcţionalitate trebuie extinsă pentru a ţine cont şi de atributele de calitate ale serviciilor alese. Criteriile QoS constituie de mai mulţi ani o preocupare majoră în cadrul aplicaţiilor de timp real şi a reţelelor, dar abia recent a început să fie abordată şi problema integrării QoS în cadrul workflow-urilor [**12**].

În general, pentru fiecare serviciu abstract din cadrul unui proces există o serie de servicii concrete care implementează funcţionalitatea dorită. Un sistem de compunere automată a serviciilor trebuie să selecteze câte un serviciu concret din lista de servicii candidat asociată fiecărui serviciu abstract, astfel încât serviciul compus obţinut să aibă calitatea optimă. În cazul selecției serviciilor pe baza a mai mult de doi parametri de calitate diferiţi, avem de a face cu o problemă de optimizare multicriterială. Astfel de probleme sunt NP-complexe, iar pentru rezolvarea lor sunt folosiți cel mai adesea algoritmi euristici. În cadrul tezei sunt prezenate cele mai întâlnite astfel de abordări.

Pentru a putea optimiza selecţia serviciilor concrete care împreună oferă o compunere de servicii conformă criteriilor clientului, trebuie găsită o metodă de calcul a valorii QoS a serviciului compus. Pentru rezolvarea acestei probleme există două abordări principale. Cea dintâi constă în efectuarea de simulări pentru determinarea valorii QoS a serviciului compus. O astfel de abordare este descrisă în [**13**]. Cea de a doua abordare, pe care o vom prezenta pe larg în secţiunea următoare, constă în utilizarea de formule matematice pentru calcularea valorilor QoS agregate.

2.4.2 Agregarea atributelor QoS

Agregarea atributelor QoS este necesară pentru a calcula QoS global al unei compuneri de servicii plecând de la atributele QoS ale serviciilor participante la compunere. Estimarea valorii globale QoS este foarte importantă pentru clienții unei compuneri de servicii. Pentru a determina valoarea globală QoS este nevoie ca toate atributele QoS ale serviciilor implicate să fie descrise folosind un limbaj comun. De obicei însă, descrierile QoS ale diferitelor servicii sunt eterogene, fapt ce îngreunează estimarea unei valori globale. Atributele QoS au un format diferit, în funcție de natura mărimii considerate (spre ex. numeric sau procent în cazul unei mărimi statistice, sau text).

O serie de contribuții ştiințifice adresează această problemă și încearcă stabilirea unui set de funcții matematice pentru a putea calcula valoarea globală QoS a unei compuneri [**11**]. Unele dintre acestea [**12**][**14**] abordează calcularea valorii QoS agregate pe baza unui scenariu, pentru un set de parametri predefiniți cum ar fi timpul de răspuns sau costurile. În aceste cazuri, modelul de compoziție nu definește structuri generale, iar scenariile sunt limitate pentru exemplele descrise.

Lucrarea [**15**] definește conceptele de bază ce stau la baza unei compuneri și analizează şabloanele de compunere existente în workflow-uri. Sunt stabilite ca fiind de bază șapte şabloane de compunere: secvență, iterație, XOR-XOR, AND-AND, AND-DISC, OR-OR și OR-DISC. Plecând de la ideea de bază că orice compunere de servicii web poate fi prezentată ca fiind o combinație a acestor şabloane și folosind funcțiile care calculează valoarea QoS pentru fiecare dintre ele, se poate calcula valoarea QoS agregată. Fluxul activităților din cadrul compunerii de servicii este modelat ca un graf, iar acest graf este transformat într-un graf de şabloane de compunere definite anterior.

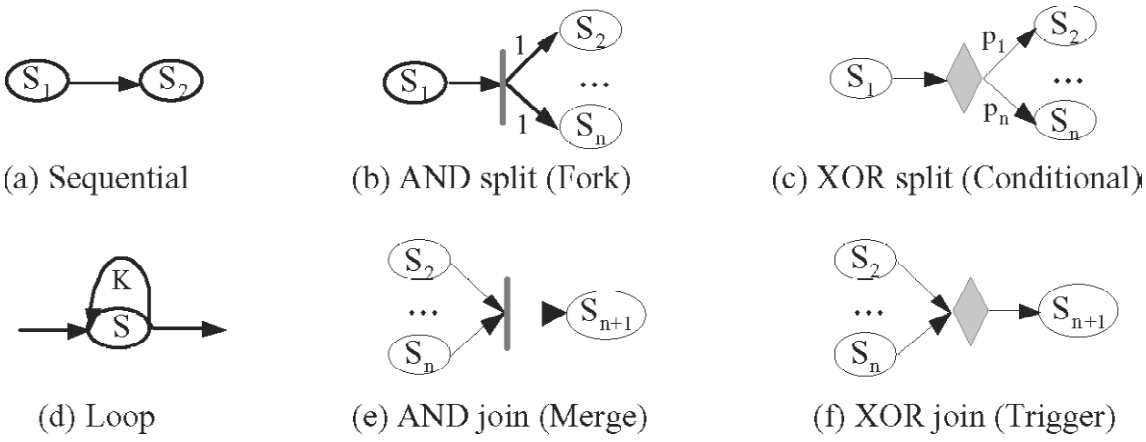

Fig. 1 Şabloane de compunere a serviciilor web

Pentru un şablon de tipul XOR, fiecărei posibilităţi de execuţie îi este asociată o probabilitate specificată de către client, iar valoarea finală a atributului pentru această secvenţă este calculată folosind această probabilitate. Pentru o secvenţă de tip Loop este necesară

definirea unui probabilităţi k de execuţie a operaţiei astfel încât calculul final QoS este produsul dintre probabilitatea k şi valoarea atributului pentru o operaţie.

În funcţie de natura unui atribut QoS, calculul valorii QoS totale se face folosind diferite calcule matematice. Spre exemplu, în cazul unei secvenţe, un atribut de tipul cost va fi calculat adunând costurile tuturor serviciilor implicate, în timp ce disponibilitatea unui sistem va fi calculată ca fiind produsul disponibilităţiilor sistemelor implicate. Modul de calcul al valorii agregate a unui anumit atribut QoS trebuie în general să fie specificat de client, dar pentru atribute frecvent utilizate, o aplicaţie ar putea sugera o metodă de calcul.

Lucrarea [**16**] defineşte următoarea tabelă de calcul a valorilor agregate QoS pentru atributele QoS frecvent întâlnite. Formulele de agregare au la bază metoda de calcul introdusă în [**17**].

QoS Attr.	Sequence	Switch	Flow	Loop
Time (T)	$\sum_{i=1}^{m} T(t_i)$	$\sum_{i=1}^{n} p_{ai} * T(t_i)$	$Max\{T(t_i)_{i\in\{1\dots p\}}\}$	$k * T(t)$
Cost (C)	$\sum_{i=1}^{m} C(t_i)$	$\sum_{i=1}^{n} p_{ai} * C(t_i)$	$\sum_{i=1}^{p} C(t_i)$	$k * C(t)$
Availability (A)	$\prod_{i=1}^{m} A(t_i)$	$\sum_{i=1}^{n} p_{ai} * A(t_i)$	$\prod_{i=1}^{p} A(t_i)$	$A(t)^k$
Reliability (R)	$\prod_{i=1}^{m} R(t_i)$	$\sum_{i=1}^{n} p_{ai} * R(t_i)$	$\prod_{i=1}^{p} R(t_i)$	$R(t)^k$
Custom Attr. (F)	$f_S(F(t_i))$ $i \in \{1 \dots m\}$	$f_B((p_{ai}, F(t_i)))$ $i \in \{1 \dots n\}$	$f_F(F(t_i))$ $i \in \{1 \dots p\}$	$f_L(k, F(t))$

Tab. 1 Funcţii de agregare ale atributelor QoS [16]

O altă categorie o reprezintă atributele QoS de tip boolean pentru care valoarea agregată poate fi calculată folosind un operator XOR. În acest caz, atributului QoS îi poate fi asociată o valoare întreagă (0=false, 1=true), valoarea QoS agregată putând fi calculată ca produsul acestor valori întregi. Din această categorie fac parte atributele de securitate, cum ar fi autentificarea, criptarea, confidenţialitatea.

Valoarea QoS agregată a unui serviciu compus poate fi calculată recursiv, prin agregarea atributelor QoS calculate la fiecare nivel de şabloanele de compunere.

2.4.3 Calculul valorii QoS agregate în cazul serviciilor compuse

Abordările existente pentru estimarea valorii QoS agregate în cazul serviciilor compuse propun metode care diferă prin restricţiile impuse tipologiei compunerii. Majoritatea metodelor actuale se limitează la modele de orchestrare reprezentate de workflow-uri structurate.

Yang et al. [**5**] au introdus o metodă care depăşeşte aceste limitări, oferind posibilitatea de calculare a valorii QoS agregate pentru workflowuri nestructurate. Această metodă primeşte ca parametri de intrare un model de orchestrare şi pentru fiecare activitate din model un serviciu concret correspondent. Modelul de orchestrare este un graf orientat care ataşează

probabilităţi de execuţie nodurilor componente. Suma probabilităţilor de pe muchiile ce pornesc dintr-o poartă XOR este 1. Toate celelalte muchii au asociată probabilitatea 1.

Modelele de orchestrare sunt decompuse în *componente de orchestrare* care sunt reprezentate de subgrafuri cu un singur punct de intrare şi un singur punct de ieşire. Valoarea QoS agregată este calculată folosind o metodă bottom-up pentru fiecare componentă de orchestrare.

Modelele de orchestrare structurate, în care fiecare poartă SPLIT are o poartă JOIN corespondentă, pot fi uşor analizate. În funcţie de tipul parametrului QoS există diferite formule de agregare,care pot fi împărţite în trei categorii:

1. **Cale critică** - Valoarea QoS agregată este determinată de calea critică a modelului de orchestrare. Exemple: *timpul de execuţie*, pentru care se foloseşte calea critică cea mai lungă; *toleranţa la defecte*, pentru care se foloseşte calea critică cea mai scurtă.
2. **Aditiv** - Valoarea QoS agregată este suma ponderată a valorilor QoS ale serviciilor componente, folosind ponderi proporţionale cu frecvenţa cu care fiecare serviciu este invocat. Exemplu: *costul*.
3. **Multiplicativ** - Valoarea QoS agregată este produsul valorilor QoS ale serviciilor componente, aplicând fiecărei valori QoS un exponent proporţional cu frecvenţa cu care acesta este invocat. Exemplu: *fiabilitatea*.

Un pas premergător agregării QoS este folosirea tehnicii de structurare introduse în [**18**] pentru a transforma un model de orchestrare nestructurat într-un model de orchestrare maxim-structurat.

Componentele care în urma folosirii acestei metode ramân ireductibile se numesc *componente rigide* şi sunt de două tipuri: *grafuri orientate aciclice* (DAG - Acyclic Graphs) şi bucle ireductibile de tipul *intrări-multiple - ieşiri-multiple* (MEME - multiple-entry, multiple-exit). Autorii lucrării [**5**] oferă un algoritm care transformă componentele DAG ireductibile în componente de decizie echivalente. Buclele MEME ireductibile pot fi transformate în componente rigide echivalente, unde concurenţa este complet încapsulată în subcomponente. Pentru aceste componente echivalente se poate calcula de câte ori un nod din cadrul buclei MEME va fi vizitat. Astfel valoarea QoS a componentei ireductibile poate fi calculată folosind formulele de agregare caracteristice categoriei atributului QoS.

Importanţa calculării automate a valorii globale QoS a unui sistem este dată atât de dorinţa de a oferi o soluţie de compunere iniţială optimă, cât şi de a putea recalcula o nouă compunere de servicii în cazul în care unul dintre serviciile componente devine indisponibil, ţinând cont de caracterul imprevizibil al sistemelor reale.

2.5 Evaluarea bazată pe criterii multiple

În prezent, există o ofertă foarte mare de servicii web care îndeplinesc funcționalități identice sau similare. Designerul unei aplicații web se confruntă cu problema complicată a

alegerii serviciului web care corespunde cel mai bine criteriilor unui utilizator având la dispoziție o multitudine de alternative. Pentru rezolvarea acestei probleme uzuale un rol esențial îl au analiza și compararea caracteristicilor nefuncționale QoS (Quality of Service) ale serviciilor web.

În practică există în mod curent situații în care trebuie urmărite mai multe obiective, de obicei conflictuale, de aceea selecţia serviciilor web bazată pe proprietăţile nefuncţionale reprezintă o problemă de optimizarea multiobiectiv. Un exemplu este oferit în tabela de mai jos, în care se consideră un serviciu web care trebuie să ofere un timp de răspuns mic, indisponibilitate a sistemului redusă, precum şi un preţ scăzut.

Atribut QoS	WS1	WS2	WS3	WS4
Preţ	30	32	30	31
timp de răspuns	7	8	6	3
indisponibilitate	1%	2%	2%	1%

Tab. 2 Exemplu servicii web candidat

Aşa cum se observă, toate atributele web ale serviciului WS2 au valori inferioare celorlalte servicii, în timp ce pentru celelalte servicii nu există o relaţie de superioritate clară. Folosind terminologia specifică optimizării multiobiectiv, spunem că soluţia WS2 este *dominată* de celelalte soluţii, în timp ce WS1, WS3 şi WS4 poartă numele de soluţii *nedominate* sau soluţii *Pareto optimale*. Mulţimea tuturor soluţiilor nedominate formează aşa numitul *front Pareto* al unei probleme multiobiectiv. Deoarece niciuna dintre soluţiile de pe frontul Pareto nu este superioară celorlate, alegerea uneia dintre acestea se face pe baza preferinţelor clientului.

2.6 Luarea deciziei

Tehnicile de optimizare bazate pe criterii multiple se pot clasifica în funcţie de momentul în care se iau în considerare preferinţele în cadrul căutarii şi luării deciziei privind soluţia optimă aleasă, astfel:

- Tehnici în care preferinţele sunt enunţate *a-priori* presupun ca înainte de a efectua optimizarea preferinţele să fie analizate şi agregate într-o singură funcţie / un singur obiectiv de către factorul de decizie. Această abordare transformă o problemă multi-obiectiv într-o problemă cu un singur obiectiv.
- Tehnici în cadrul cărora preferinţele sunt enunţate în mod progresiv şi sunt integrate în procesul de căutare. Aceste tehnici sunt considerate a fi interactive iar preferinţele sunt aplicate pe măsura ce se efectuează căutarea. Factorul de decizie interacţionează cu programul de optimizare în timpul procesului de optimizare. În cadul acestor tehnici, pe parcurs sunt oferite seturi de soluţii noi, iar factorul de decizie hotărăşte cum sunt schimbate criteriile de căutare.
- Tehnici în cadrul cărora preferinţele sunt enunţate a-posteriori. Căutarea se face înainte de luarea unei decizii, algoritmul de optimizare oferă un set de soluţii potrivite, din spaţiul Pareto optim, dintre care factorul de decizie alege soluţia optimă.

2.7 Metode de rezolvare

2.7.1 Metode tradiţionale de exprimare a preferinţelor

Din cadrul tehnicilor a-priori fac parte ordonarea lexicografică, ordonarea min-max, abordarea neliniară, abordarea bazată pe sumă ponderată, folosind agregarea obiectivelor. Aceste metode transformă problema multicriterială într-o problemă liniară obișnuită.

2.7.1.1 Metoda sumei ponderate

Abordarea bazată pe suma ponderată este des întâlnită în practică, fiind o metodă uşor de implementat. Fiecărei funcţii obiectiv îi este asociată o pondere w_i unde $\sum w_i = 1$.

$$y = f(x) = w_1 \cdot f_1(x) + w_2 \cdot f_2(x) + \ldots + w_k \cdot f_k(x))$$

Metoda are însa o serie de dezavantaje: nu este însă capabilă să ofere soluţii din zona concavă a frontului Pareto; nu oferă o distribuţie uniformă a soluţiilor în cadrul setului optimal Pareto; este greu de diferenţiat între setarea ponderilor pentru a compensa diferenţele între magnitudinile funcţiilor obiectiv şi setarea ponderilor pentru a indica importanţa relativa a obiectivelor. O analiză amplă a acestei abordări este prezentată în [**19**], unde autorii concluzionează că metoda sumei ponderate este în mod fundamental incapabilă să exprime preferinţe complexe.

2.7.1.2 Ordonarea lexicografică

Ordonarea lexicografică consideră un set de *n* criterii asupra căruia se aplică o ordine liniară. Se consideră că o alternativă precede lexicografic o altă alternativă dacă are o valoare mai bună pentru primul criteriu a cărui valoare diferă. Metoda lexicografică produce o ordonare strictă a aletrnativelor, dar are dezavantajul major că este necompensatoare.

Pentru exemplul prezentat în Tab. 2 , considerând preţul ca fiind cel mai important criteriu alegem serviciile WS1 şi WS3 ca fiind cele mai bune şi pentru a le departaja, le comparăm conform următorului atribut, timpul de răspuns. Astfel serviciul WS3 este serviciul câştigător, având timpul de răspuns mai mic.

2.7.1.3 Ordonarea semilexicografică

O extensie a ordonării lexicografice este ordonarea semilexicografică. Aceasta permite stabilirea unor compromisuri în cazul în care o îmbunatățire majoră a unui criteriu poate compensa o depreciere minoră a criteriului principal. Se consideră că o alternativă *x* este mai bună decât o alternativă *y* în cazul în care criteriul cel mai important care stabileste această ordonare este mai mare pentru alternativa *x* față de alternativa *y* cu o valoare care depășeste un prag fix stabilit. Această metodă previne alegerea unei alternative ca fiind cea mai bună în cazul în care aceasta este doar nesemnificativ mai bună pentru criteriul principal, dar considerabil mai proastă pentru celelalte criterii. Dezavantajul major al acestei metode este că poate conduce la relaţii intranzitive între preferinţe.

Pentru exemplul prezentat în Tab. 2 , considerăm preţul ca fiind cel mai important criteriu, iar valoarea de prag fix este 3. Astfel o valoare mai mică de 3 a preţului nu este relevantă atât timp cât permite obţinerea unei valori considerabil mai bune pentru unul dintre celelalte două atribute. WS1 şi WS3 au cel mai bun preţ (30) dar WS4 are un preţ doar puţin mai mare (31) în schimb un timp de răspuns considerabil mai bun. De aceea, folosind metoda semilexicografică el va fi desemnat cel mai bun dintre serviciile web disponibile.

2.7.2 Algoritmi evolutivi de optimizare

O alternativă recentă la metodele clasice pentru rezolvarea problemelor multicriteriale este folosirea de algoritmi evolutivi. Algoritmii evolutivi folosesc populaţii ce oferă mai multe soluţii de compromis potenţiale, într-o singură rulare. De asemenea, algoritmii evolutivi pot lucra cu spaţii de căutare mari şi complexe.

Algoritmii evolutivi reprezintă o clasă de metode de optimizare stohastică ce simulează procesul natural de evoluţie. Există numeroase metodologii evolutive, de bază fiind algoritmii genetici, programarea şi strategiile evoluţioniste. Algoritmii au la bază o mulţime de soluţii candidat ce sunt modificate succesiv folosind două principii de bază ale evoluţiei: selecţia si variaţia.

Mecanismul de selecţie alege soluţiie cele mai bune, soluţiile care au probabilitatea cea mai mare să se reproducă. În cadrul algoritmilor evolutivi, pentru a alege soluţia cea mai bună este simulat un proces de selecţie stohastic în care fiecare soluţie se poate reproduce de un număr de ori, în funcţie de calitatea acesteia. Prin evaluarea indivizilor populaţiei şi atribuirea de valori de fitness scalare acestora se poate compara calitatea lor.

Variaţia, cel de-al doilea principiu de bază, simulează fenomenul natural de creare de noi indivizi folosind recombinări şi mutaţii.

Cu toate că principiile ce stau la baza algoritmilor evolutivi sunt simple, algoritmii oferă o metodă de căutare puternică [**20**]. Algoritmii evoluţionişti sunt în mod deosebit potriviţi pentru rezolvarea problemelor de optimizare multicriterială datorită capacităţii lor de a genera multiple soluţii Pareto optimale într-o singură rulare si a posibilităţii de recombinare a acestor soluţii pentru a obţine soluţii superioare.

În practică metodele bazate pe algoritmi genetici trebuie adaptate la funcţiile obiectiv, constrângerile şi codificarea problemei date, iar setul de soluţii Pareto găsite de algoritmii genetici trebuie minimizat pentru ca designerul aplicaţiei să poată alege cea mai bună variantă dintr-un set mic de soluţii optime. O astfel de variantă nu este eficientă pentru compunerea dinamică a serviciilor web, deoarece are nevoie de inverventţie umană.

3. Metoda condiţional lexicografică QoSPref

În cadrul unei probleme de selecţie având obiective multiple, distingem între obiective care trebuie obligatoriu îndeplinite şi obiective opţionale. Astfel, serviciul ales trebuie să îndeplinească cerinţele obligatorii iar în cazul în care niciun serviciu nu îndeplineşte toate preferinţele opţionale ale clientului, trebuie ales cel mai bun compromis de preferinţe opţionale conform cerinţelor acestuia. În acest scop, avem nevoie de un limbaj în care clientul să îşi poată descrie preferinţele şi prioritatea lor şi de un mecanism de decizie care să poata interpreta cerinţele clientului şi să aleagă soluţia cea mai potrivită conform cerinţelor specificate, în mod automat, dinamic.

Metoda propusă se bazează pe modul în care raţionează oamenii aflaţi în situaţia de a alege o variantă dintre o mulţime de posibilităţi. În mod uzual oamenii îşi crează un set de reguli care să îi ajute să analizeze şi să compare variantele existente. Prioritizarea preferinţelor este similară cu impunerea unei ordini lexicografice asupra setului de preferinţe, dar în practică o ordonare strictă este insuficientă pentru a descrie complet cerinţele existente. Astfel, oamenii stabilesc un set de reguli care schimbă prioritatea preferinţelor atunci când anumite condiţii sunt îndeplinite.

Pentru descrierea preferinţelor, metoda propusă pune la dispoziţie o notaţie simplă, care oferă posibilitatea ataşării de reguli la preferinţele lexicografice. Notaţia oferă un limbaj intuitiv, care nu necesită cunoştinte tehnice, astfel putând fi folosit de un număr mare de clienţi. Metoda foloseşte un algoritm care analizează atât priorităţile preferinţelor, cât şi regulile ataşate lor şi stabileşte o ordine totală a alternativelor.

3.1 Notaţie pentru descrierea preferinţelor

Notaţia propusă oferă posibilitatea descrierii preferinţelor clientului. Preferinţele QoS se definesc în două blocuri separate: un bloc care conţine constrângerile obligatorii şi un bloc în care sunt definite preferinţele opţionale şi eventual condiţiile asociate lor.

Pentru a exemplifica, presupunem exemplul unei companii care oferă servicii de vizualizare a datelor, cum ar fi generarea de grafice pe baza unui set de date. Compania deleagă generearea graficelor altor firme externe, care oferă servicii web pentru generarea de grafice. Un client îşi specifică preferinţele folosind această notaţie iar pe baza acestei specificaţii şi a algoritmului descris în secţiunea 3.2 un *web service broker* alege serviciul care oferă cea mai potrivită funcţionalitate pentru cererile clientului. Serviciile web care oferă funcţionalitatea de generare de grafice au atât proprietăţi QoS independente de domeniul de activitate, cum ar fi timpul de răspuns sau disponibilitatea, cât şi proprietăţi din cadrul domeniului de activitate, cum ar fi tipul de grafic generat, preţul solicitat pentru generarea unui grafic sau rezoluţia graficelor generate.

Considerăm drept client al acestei companii un sistem de achiziţii de date care afişează în mod regulat (spre exemplu, la intervale de 5 secunde) un grafic care prezintă situaţia unui

proces de producţie. Graficul este vizualizat pe un monitor având o rezoluţie de 1280x780 pixeli, care suportă maxim 65536 culori. Graficul este actualizat la fiecare 5 secunde, iar în cazul în care acesta nu este disponibil, este omisă o actualizare iar următoarea afişare are loc după încă un interval, adică abia după 10 secunde.

Constrângerile sunt definite în cadrul unui bloc "*constraints*", sub forma unei liste, în care fiecare intrare constituie o condiţie booleană care trebuie să fie îndeplinită în mod obligatoriu de către serviciu. Ordinea în care sunt definite constrângerile în cadrul listei nu este importantă, deoarece toate constrângerile trebuie îndeplinite.

Pentru exemplul ales de noi extragem urmatoarele constrângeri: clientul are nevoie de o vizualizare a datelor folosind grafice de tipul "time series", costul serviciului trebuie să nu depăşească 10 unităţi (de ex. Euro), serviciul trebuie să fie disponibil în proporţie de 95%, rezoluţia necesară este de 1280x720 pixeli, iar timpul de răspuns trebuie să fie mai mic decât 10 unităţi (de ex. secunde). Considerăm următorea definiţie de constrângeri, pentru exemplul ales:

```
constraints {
        chartType = "time series",
        cost < 10,
        availability > 0.95,
        imageResolution = "1280x720",
        responseTime < 10
}
```

Clientul îşi defineşte preferinţele opţionale asupra serviciului sub forma unei liste, în cadrul blocului de preferinţe "*preferences*". În timp ce ordinea cerinţelor nu este relevantă pentru constrângerile obligatorii, pentru preferinţe ordinea este importantă. Preferinţele trebuie specificate în ordinea importanţei, pe primul loc fiind definită cea mai importantă preferinţă considerată.

```
preferences {
        cost,
        availability : high,
        responseTime,
        colors : high
}
```

Pentru fiecare preferinţă definită clientul trebuie să indice direcţia în care se găsesc valorile optimale, spre exemplu, un preţ este considerat cu atât mai bun, cu cât valoarea lui este mai mică, în timp ce disponibilitatea serviciului este cu atât mai bună, cu cât valoarea sa este mai mare. Valorile posibile sunt "*low*" si "*high*", în funcţie de direcţia care indică valorile mai bune. Operatorul default, care este considerat în cazul în care nu este specificată nici o direcţie, este operatorul "*low*", adică valorile mai mici sunt preferate. Astfel, în definiţia de mai sus, preferinţele "preţ" (*price*) şi "timp de răspuns" (*responseTime*) folosesc opreratorul

default, având direcţia “*low*”. Datorită ordinii specificate, preferinţa “preţ” este considerată ca fiind cea mai importantă preferinţă.

Preferinţele specificate în forma actuală nu reflectă în mod corect şi complet cerinţele clientului. Metoda propusă stabileşte în mod automat o ordine totală a alternativelor şi are nevoie de definirea într-un mod complet a preferinţelor clienţilor în notaţia aleasă pentru a putea stabili serviciul web câştigător.

O primă problemă pe care o putem considera este situaţia în care timpul de răspuns al serviciului depăşeşte 5 secunde. În acest caz, o actualizare a graficului ar fi omisă, situaţie care trebuie evitată chiar dacă trebuie considerat un preţ mai mare. Constrângerea definită de client legată de preţ trebuie respectată, dar o diferenţă de preţ între două servicii considerate care oferă timpuri diferite de răspuns trebuie să poată fi specificată. O diferenţă a timpului de răspuns este relevantă pentru client numai în condiţiile în care un serviciu web oferă un timp de răspuns mai mare decât cele 5 secunde, iar celălalt un timp de răspuns mai mic. În cazul in care ambele servicii au timpi de răspuns diferiţi, dar mai mici decât 5 secunde, valoarea exactă a timpului de răspuns nu contează, pentru că actualizarea graficului se face doar după 5 secunde, astfel încât un timp de răspuns mai mic nu aduce niciun avantaj practic.

O altă situaţie care trebuie considerată este diferenţa de calitate a graficului rezultat, datorată numărului de culori. În cazul în care un serviciu oferă grafice având un număr mai mic de culori decât 65536, calitatea graficului nu este optimală. Pe de altă parte, o diferenţă a numărului de culori pentru valori mai mari decât numărul maxim de culori suportat de monitorul clientului, nu este sesizabilă pentru acesta, deci nu trebuie sa fie considerată relevantă. Pentru valori nesatisfacătoare ale acestui parametru, o diferenţă mică a preţului trebuie acceptată în cazul în care duce la o creştere a calităţii imaginii graficului generat.

Pentru a putea specifica aceste condiţii din cadrul exemplului descris, notaţia propusă oferă patru operatori unari care sunt definiţi în tabela următoare:

Operator	Semnificaţie
AT_LEAST_ONE* (condition)	condition($service_1$) OR condition($service_2$)
EXACTLY_ONE (condition)	condition($service_1$) XOR condition($service_2$)
ALL (condition)	condition($service_1$) AND condition($service_2$)
DIFF (attribute)	|$service_1$.attribute - $service_2$.attribute|

Fig. 2. Operatori oferiţi de notaţie

Primii trei opreratori necesită ca parametru o expresie booleană care este evaluată de două ori, pentru ambele servicii web care sunt comparate. Expresia booleană poate avea ca parametri unul sau mai multe atribute QoS. Operatorii booleeni OR, XOR, AND evaluează rezultatele expresiilor booleene corespunzătoare serviciilor comparate şi oferă rezultatul condiţiei exprimate.

Ultimul operator, DIFF, necesită ca parametru un atribut QoS şi oferă ca rezultat modulul diferenţei dintre valorile acestui atribut pentru serviciile comparate.

Operatorul AT_LEAST_ONE este definit ca operator default şi nu trebuie specificat explicit.

Folosind aceşti operatori, cerinţele clientului descrise în exemplul de mai sus pot fi formulate folosind notaţia noastră în felul următor:

```
preferences {
        [EXACTLY_ONE(responseTime >5)] responseTime,
        [DIFF(cost) > 2] cost,
        [colors < 65536] colors : high,
        cost,
        availability : high,
        responseTime,
        colors : high
}
```

Fig. 3. Definiţia preferinţelor clientului

Condiţia $[\text{colors} < 65536]\ \text{colors} : \text{high}$ nu are definit în mod explicit un operator aşa că este folosit operatorul default AT_LEAST_ONE.

3.2 Algoritm pentru ordonarea alternativelor

Pentru a afişa rezultatul comparării a două servicii s1 şi s2 introducem următoarea notaţie:

- $s_1 \succ s_2$ serviciul web s_1 este preferat serviciului s_2
- $s_1 \prec s_2$ serviciul web s_2 este preferat serviciului s_1
- $s_1 \sim s_2$ între serviciile web s_1 şi s_2 există o relaţie de indiferenţă (nu putem spune că unul dintre servicii este preferat celuilalt)
- $s_2 \succ_k s_1$ serviciul web s_1 este preferat serviciului web s_2 (condiţia *k* din blocul de preferinţe a fost cea care a stabilit această ierarhie)
- $s_1 \prec_k s_2$ serviciul web s_2 este preferat serviciului web s_1 (condiţia *k* din blocul de preferinţe a fost cea care a stabilit această ierarhie)

Algoritmul folosit pentru a compara serviciile web pe baza notaţiei definite în 3.1 este enunţat în pseudocod în Fig. 4.

```
1.  function compareServices(service1, service2, preferences)
2.    for i ← 1 .. length(preferences) do
3.      cond ← preferences[i].condition
4.      attr ← preferences[i].attribute
5.      dir ← preferences[i].direction
6.      if cond = null OR cond(service1, service2) = true then
7.        result ← compare(service1.attr, service2.attr, dir)
8.        if result ≠ 0 then
9.          return {result, i}
10.       end if
11.     end if
12.   end for
13.   return null
14. end function
```

Fig. 4. Algoritmul de comparare a două servicii web pe baza notaţiei QoSPref

Algoritmul analizează toate condiţiile din cadrul blocului de preferinţe, în ordinea în care au fost enunţate, adică în ordinea importanţei lor. În cazul în care condiţia analizată nu are ataşat nici un operator, sau operatorul este evaluat pozitiv (linia 6), valorile atributului QoS specificat în cadrul acestei condiţii sunt evaluate pentru serviciile comparate folosind următoarea funcţie:

```
function compare(attr1, attr2, direction)
        if attr1 = attr2 then
                result ← 0
        else if attr1 < attr2 then
                result ← 1
        else
                result ← -1
        end if
        if direction = high then
                result ← -result
        end if
                return result
end function
```

Funcţia compară valorile atributelor celor două servicii web şi returnează o valoare numerică pozitivă în cazul în care primul argument este mai bun, negativă dacă cel de-al doilea argument este mai bun sau 0 dacă argumentele funcţiei sunt egale.

În cazul în care valorile atributului comparat sunt egale, algoritmul evaluează condiţia următoare din cadrul blocului de preferinţe. În momentul în care se constată o diferenţă, se returnează rezultatul comparaţiei şi numărul condiţiei care a dus la stabilirea acestei ierarhii. Stabilirea regulii care a dus la realizarea unei clasificări a serviciilor este necesară pentru a elimina eventualele intranzivităţi ale preferinţelor şi a stabili o ordine totală a serviciilor, ţinând cont de prioritaea preferinţelor. Compararea perechilor de servicii nu este suficientă pentru a stabili o ordine totală, deoarece metoda acceptă preferinţe intranzitive. Matematicianul psiholog Amos Tversky a demonstrat într-o serie de experimente faptul că

oamenii au preferinţe intranzitive, de aceea este importantă acceptarea acestor preferinţe de către metoda propusă.

Pentru a exemplifica situaţia intranzivităţii preferinţelor, considerăm următoarele cinci servicii web din Tab. 3, care au specificate trei atribute de calitate QoS.

	WS_1	WS_2	WS_3	WS_4	WS_5
responseTime	7.0	7.0	5.5	4.5	7.5
Cost	4.0	5.0	6.5	8.0	7.5
Colors	256	256	256	65536	65536

Tab. 3. Exemplu de servicii web având specificate atributele QoS

Dintre serviciile web trebuie aleasă cea mai potrivită alternativă conform preferinţelor enunţate de client în Fig. 3 în cadrul definiţiei notaţiei. Toate perechile de servicii web sunt comparate, folosind algorimul enunţat anterior, rezultatul fiind afişat în tabela Tab. 4. Rezulatul comparării unei perechi de servicii web WS_i şi WS_j este afişat în tabelă în dreptul simbolului corespunzător i/j.

½	1/3	1/4	1/5	2/3	2/4	2/5	3/4	3/5	4/5
$\sim$	$\succ_2$	$\prec_1$	$\succ_2$	$\prec_4$	$\prec_1$	$\succ_2$	$\prec_1$	$\prec_3$	$\succ_1$

Tab. 4. Rezultatele comparaţiei perechilor de servicii web

Analizând rezultatul comparaţiei se pot observa următoarele cazuri de intranzitivitate a preferinţelor:

$WS_1 \succ WS_3 \succ WS_2$
$WS_2 \sim WS_1$

Astfel, deşi conform comparaţiei perechilor de servicii WS_1 cu WS_3 şi WS_3 cu WS_2 reiese că serviciul WS_1 este superior serviciului WS_2, în momentul comparaţiei directe, WS_2 este echivalent ca preferinţe cu WS_1.
Un alt exemplu de intrazitivitate este următorul:

$WS_2 \prec WS_3 \prec WS_5$
$WS_5 \prec WS_2$

Pentru a stabili o ordine totală a alternativelor, definim un algoritm care ataşează fiecărui serviciu web *i,* un vector de valori întregi $V_i \in \mathbb{N}^{r+1}$, unde *r* reprezintă numărul total de preferinţe. Următorul algoritm calculează vectorii de scor ai fiecărui serviciu web comparat. Cu *n* este notat numărul de servicii web care sunt comparate.

procedure createScoreVectors ()
    for i ← 1 .. n **do**
        for k ← 1 .. r **do**
            V_i^k ← *numărul de cazuri în care serviciul WS_i este preferat unui alt serviciu web datorită regulii k (i.e., datorită unei relaţii $\succ_k$).*
        end for
        V_i^{r+1} ← *numărul de cazuri în care serviciul WS_i este într-o relaţie de indiferenţă cu un alt serviciu web.*
    end for
end procedure

Folosind acest algoritm calculăm pentru cele 5 servicii web din exemplul anterior următorii vectori de scor:

	$\succ_1$	$\succ_2$	$\succ_3$	$\succ_4$	$\sim$
WS_1	0	2	0	0	1
WS_2	0	1	0	0	1
WS_3	0	0	0	1	0
WS_4	4	0	0	0	0
WS_5	0	0	1	0	0

Fig. 5. Vectori de scor

Pentru a compara vectorii de rezultate obţinuţi folosim următorul algoritm, care stabileşte o ordine totală peste mulţimea alternativelor existente. Pentru fiecare pereche de servicii web se stabileşte si se compară numărul total de cazuri în care un serviciu web este preferat altor servicii web. Serviciul care este mai des preferat este ales ca fiind câştigător. În cazul în care acest prim pas nu este suficient pentru a determina cel mai bun dintre cele două servicii ale unei perechi, se compară de câte ori un serviciu este preferat altor servicii datorită celei mai importante preferinţe (prima intrare din vectorul de scor) şi astfel se stabileşte serviciul câştigător. În cazul în care nici acest pas nu este suficient, se trece la următoarea intrare din vectorul de scor şi procedeul se repetă până când se poate desemna un serviciu web câştigător. În cazul în care vectorii de scor sunt identici, algoritmul returnează valoarea 0, serviciile web fiind echivalente ca prioritate. Algoritmul este descris în pseudocod în Fig. 6.

Folosind acest algoritm pentru exemplul de servicii web din Tab. 3 ajungem la următoarea clasificare: (WS_4, WS_1, WS_2, WS_5, WS_3), unde serviciul WS_4 este considerat a fi cea mai potrivită alternativă pentru cerinţele date. Astfel algoritmul înlătură clasificarea ciclică, stabilind o ordine totală chiar şi în situaţia existenţei preferinţelor intranzitive.

function compareScores(V_1, V_2)

$\text{count}_1 \leftarrow \sum_{i}^{r} V_1^i$

$\text{count}_2 \leftarrow \sum_{i}^{r} V_2^i$

if $\text{count}_1 \neq \text{count}_2$ **then**
return count_1 - count_2
end if
for i ← 1 .. r + 1 **do**
if $V_1^i \neq V_2^i$ **then**
return $V_1^i - V_2^i$
end if
end for
return 0
end function

Fig. 6. Algoritmul de comparare a vectorilor de scor

Această metodă oferă posibilitatea selectării celui mai bun serviciu web dintr-o mulţime de alternative posibile, ţinând cont de preferinţele QoS exprimate de client. Avantajul principal al metodei noastre este că permite descrierea preferinţelor într-o manieră simplă şi intuitivă, care nu impune cunoaşterea unor tehnici de analiză şi decizie multicriterială. Limbajul oferit pentru specificarea preferinţelor se ghidează după modul în care oamenii raţionează şi stabilesc compromisuri în momentul în care se află în faţa unei decizii de a alege cea mai bună soluţie dintr-o mulţime de alternative. Faptul că preferinţele sunt exprimate a-priori, fără a necesita intervenţia ulterioară a clientului în procesul de decizie, face posibilă selecţia dinamică şi automată a celui mai bun servicu web.

3.3 Aplicaţie practică

Pentru a demonstra practic viabilitatea metodei noastre, oferim o implementare a motorului de selecţie dinamică a celui mai bun serviciu web dintr-o serie de alternative, pe baza abordării condiţional lexicografice propuse. Motorul de selecţie este oferit sub forma unei biblioteci Java, pe care o punem la dispoziţie sub forma unui proiect open-source denumit QoSPref, accesibil la adresa http://qospref.sourceforge.net. Un scenariu tipic de ultilizare a acestei biblioteci este ca parte integrantă a unui framework SOA, care accesează motorul de selecţie prin intermediul interfeţei de programare (API) puse la dispoziţie de biblioteca noastră. Framework-ul SOA, care joacă rol de client al motorului de selecţie, trebuie să ofere ca date de intrare informaţii legate de serviciile web candidat, de atributele QoS ale acestora, precum şi de preferinţele QoS care trebuie luate în considerare.

În plus, pentru a ilustra modul de utilizare a bibliotecii şi a uşura înţelegerea funcţionării motorului de selecţie, QoSPref oferă şi o interfaţă grafică prin intermediul căreia pot fi accesate funcţiile oferite de bibliotecă.

4. Compunerea serviciilor folosind QoSPref

În capitolul anterior am introdus limbajul QoSPref, care permite specificarea unor preferinţe QoS complexe şi am propus un algoritm de ordonare a serviciilor web în funcţie de aceste preferinţe. În continuare, ne propunem să studiem modul în care QoSPref poate fi folosit pentru selecţia serviciilor concrete care produc serviciul compus cu cele mai bune valori QoS.

Există la ora actuală o serie de framework-uri SOA capabile să realizeze compunerea dinamică a serviciilor, însă niciunul dintre acestea nu rezolvă în mod satisfăcător problema preferinţelor QoS. Cele mai multe dintre ele oferă doar posibilitatea specificării de constrângeri. Foarte puţine permit şi exprimarea preferinţelor QoS, iar acest lucru este posibil numai într-o formă rudimentară. Intenţia noastră este de a oferi o soluţie a acestei probleme, care să poată fi în principiu integrată în oricare dintre framework-urile de compunere dinamică existente. Acest lucru este important, deoarece în prezent niciunul din framework-urile existente nu s-a impus ca standard şi nu există un consens referitor la arhitectura acestora sau la modalitatea optimă de descriere semantică a caracteristicilor funcţionale şi nefuncţionale ale serviciilor web.

Procesul de compunere dinamică a serviciilor web implică trei paşi importanţi [**21**]:

1. specificarea serviciului compus
2. selecţia serviciilor web componente
3. execuţia serviciului compus

Metoda condiţional lexicografică QoSPref poate fi utilizată pentru a implementa cel de-al doilea pas al compunerii dinamice: selecţia serviciilor web componente. Fiind dat un serviciu compus descris sub forma unui workflow care specifică serviciile abstracte participante, pasul 2 are rolul de a asocia fiecărui serviciu abstract un serviciu concret care implementează funcţionalitatea cerută. Acest pas poartă şi numele de "legare a serviciilor" (service binding). În cazul ideal, serviciile concrete sunt selectate astfel încât valoarea globală QoS a serviciului compus rezultat să fie optimă.

În cadrul lucrării [**22**] autorii introduc conceptul de “Composition as a Service” (CaaS) şi prezintă modul în care această abordare a fost folosită pentru a implementa modulul de compunere a serviciilor pe care ei l-au integrat in framework-ul VRESCo. Scopul nostru nu este de a oferi o implementare completă a funcţionalităţii "Composition as a Service", ci de a oferi sub forma unui serviciu operaţia de selecţie a serviciilor web concrete care intră în componenţa unui serviciu compus. Abordarea noastră, pe care o vom denumi "Binding as a Service", are o granularitate mai mică decât "Composition as a Service". De aceea, există un număr mai mare de scenarii în care ea va putea fi folosită, iar practic orice framework de compunere dinamică a serviciilor va putea fi adaptat rapid pentru a apela servicul de selecţie oferit de noi. În acest fel, framework-ul respectiv va deveni capabil să ia în considerare preferinţe QoS complexe.

Pentru a implementa selecţia sub forma unui serviciu, va trebui să definim o interfaţă prin intermediul căreia clienţii să poată trimite cereri şi să poată primi rezultate. Aşa cum am menţionat, clienţii serviciului nostru de selecţie sunt reprezentaţi de framework-uri de compunere dinamică a serviciilor. Deoarece "Binding as a Service" corespunde pasului 2 din procesul de compunere dinamică, framework-ul client are la dispoziţie în momentul apelului specificaţia serviciului compus (construită în pasul 1). Din această specificaţie pot fi extrase în general cu uşurinţă următoarele informaţii care intră în componenţa unei cereri adresate serviciului de selecţie:

- descrierea serviciului compus sub forma unui workflow care specifică o serie de sarcini abstracte;
- pentru fiecare sarcină abstractă, o mulţime de servicii web concrete, care pot efectua sarcina respectivă;
- pentru fiecare atribut QoS, formulele de agregare corespunzătoare fiecărui şablon de compunere utilizat în descrierea workflow-ului unui serviciu compus;

În plus, o cerere adresată serviciului de selecţie trebuie să conţină şi:

- descrierea preferinţelor QoS ale clientului folosind notaţia condiţional lexicografică.

Descrierea preferinţelor QoS este singurul element al cererii care impune modificări în framework-urile de compunere existente în momentul de faţă. Celelate informaţii sunt deja disponibile şi nu necesită decât conversia la formatul impus de serviciul de selecţie, format care va fi descris într-o lucrare viitoare.

Întrucât pentru fiecare sarcină abstractă din specificaţia serviciului compus există o mulţime de servicii web candidat, operaţia de selecţie a celei mai bune combinaţii a acestora reprezintă o problemă de optimizare combinatorială multiobiectiv. Un aspect important în implementarea funcţionalităţii "selection as a service" este deci alegerea unui algoritm de optimizare performant. În secţiunea 4.3, vom analiza eficienţa unei soluţii bazate pe algoritmi genetici.

4.1 Abordarea Binding-as-a-Service (BaaS)

În cadrul compunerii serviciilor web, un punct important îl constituie selecţia serviciilor concrete asociate activităţilor din cadrul modelului de orchestrare care stă la baza modelului de compunere. În general, pentru fiecare activitate există mai multe servicii concrete care implementează funcţionalitatea cerută, dar care sunt diferite în ceea ce priveşte caracteristicile nefuncţionale, precum fiabilitatea, costul sau timpul de răspuns. De aceea, calitatea serviciului (QoS) este factorul decisiv în alegerea unui serviciu concret pentru o anumită activitate. În principiu, procesul de selecţie poate utiliza o strategie locală sau una globală. O strategie de optimizare locală presupune selectarea serviciului optim pentru fiecare activitate în parte. Complexitatea acestei strategii este polinomială, dar ea are o serie de neajunsuri, precum inabilitatea de a lua în considerare constrângeri globale între parametrii QoS. Un alt dezavantaj major este faptul că valoarea parametrilor QoS ai

serviciului compus obţinut nu este în general optimală [**23**]. Strategia preferată, pe care o vom adopta şi noi, se bazează pe abordarea planificării globale, în care valorile constrângerilor şi preferinţelor QoS sunt exprimate relativ la valorile QoS globale ale compunerii,

Astfel, primul pas necesar pentru alegerea serviciilor concrete îl constituie calcularea valorii QoS agregate a serviciului compus, pe baza atributelor QoS ale serviciilor componente. Abordările existente în acest moment se limitează la modele de compunere bazate pe workflowuri structurate. Noutatea abordării noastre BaaS este folosirea metodei de agregare propuse de Yang et al. [**5**] care depăşeşte această restricţie. Această abordare a fost descrisă în cadrul secţiunii 2.4.3.

Posibilitatea specificării unor reguli de compromis între preferinţele QoS are o importanţă majoră pentru a putea alege serviciile optime pentru un model de compunere. Abordările actuale oferă doar posibilitatea asocierii de priorităţi sau de ponderi preferinţelor clientului. Această metoda este insuficientă pentru a descrie preferinţe complexe. Noi vom integra metoda noastră QoSPref care oferă flexibilitatea exprimării unor preferinţe complexe, în cadrul compunerii serviciilor web pentru a exprima contrângerile şi preferinţele globale.

În continuare, oferim un exemplu care ilustrează problemele ce apar în cadrul compunerii dinamice a serviciilor web bazată pe calitatea serviciilor. Considerăm un sistem de trading online care oferă servicii pentru comercializarea de diverse instrumente financiare. Unele dintre serviciile oferite permit clienţilor tranzacţionarea de acţiuni domestice sau străine. Modelul care descrie procesul business este ilustrat în Fig. 7.

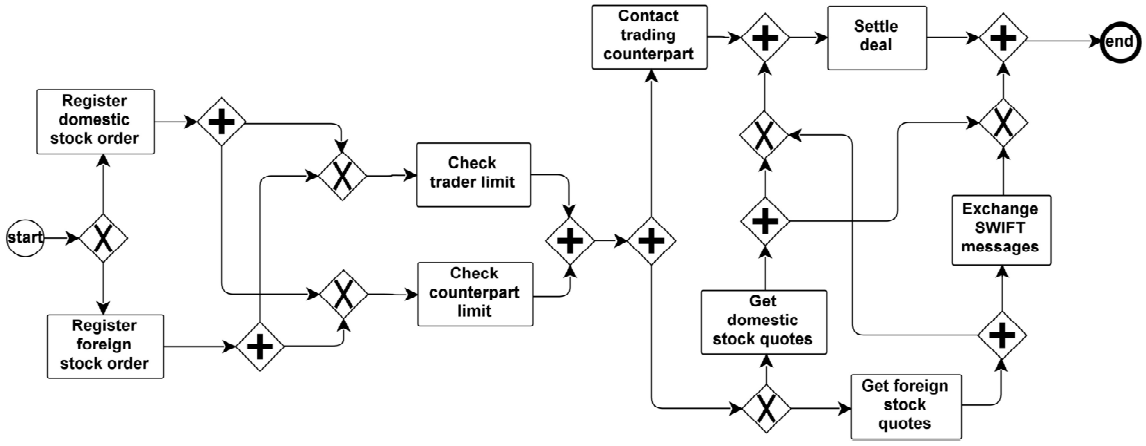

Fig. 7 Model BPMN al procesului de tranzacţionare de acţiuni bancare

Un prim pas îl constituie aducerea workflowului la o formă maxim-structurată, folosind metoda lui Yang et al. [**5**] descrisă in secţiunea 2.4.3. Modelul de orchestrare din Fig. 8 este din punct de vedere al funcţionalităţii este echivalent cu modelul prezentat în Fig. 7, dar partea din stânga a modelului este bine structurată.

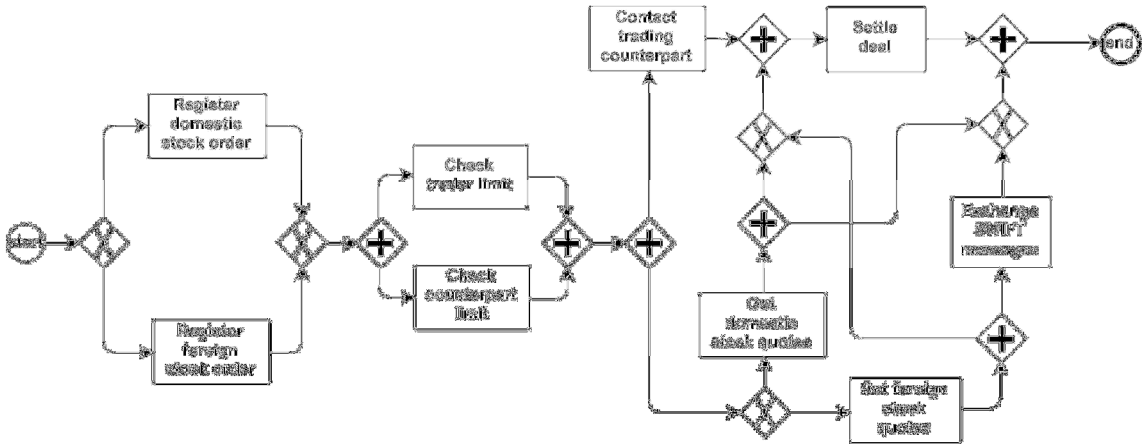

Fig. 8 Model BPMN maxim structurat al procesului de cumpărare de fonduri

Unele dintre procesele acestui acest model, cum ar fi cele pentru înregistrarea unei tranzacţii, reprezintă activităţi interne ale sistemului de tranzacţionare online. Alte procese, cum ar fi cele pentru căutarea cursurilor bancare, necesită interacţiune cu sisteme externe. Sistemul de tranzacţionare online realizează serviciul de cumpărare de fond-uri printr-un serviciu compus. În cazul proceselor care necesită interacţiune cu sisteme externe este necesar ca aceste furnizorii acestor sisteme să ofere funcţionalitatea necesară sub forma de servicii web. De obicei, pentru fiecare activitate există mai multe alternative care să implementeze funcţionalitatea cerută. Spre exemplu, există multe servicii web care să ofere cursul acţiunilor. Sistemul de tranzacţionare online trebuie să aleagă din oferta de servicii, serviciile web concrete care să implementeze fiecare funcţionalitate din cadrul modelului de compunere. Alegerea serviciilor web concrete este un process dinamic, deoarece serviciile web, fiind la rândul lor componente dinamice, pot în orice moment să dispară în timp ce noi servicii web oferind functionalitatea cerută apar.

În exemplul nostru, considerăm că singurele atribute QoS importante pentru sistemul de tranzacţionare sunt: timpul de execuţie, costul şi fiabilitatea. Valoarea QoS a serviciului compus de tranzacţionare este determinată de valorile QoS ale serviciilor componente. În timp ce costul serviciului compus poate fi calculat ca fiind suma costurilor serviciilor componente, nu există nici o metodă simplă pentru calculul timpului de execuţie şi a fiabilităţii. Pentru a putea aloca dinamic servicii concrete activităţilor serviciului compus de cumparare de fonduri, sistemul de tranzacţionare online trebuie să poată compara în mod automat valorile QoS ale serviciilor componente.

Pentru a arăta de ce realizarea unei comparaţii în cazul serviciilor compuse care au numeroase atribute QoS este o sarcină complicată, detaliem exemplul propus. Proprietarii sistemul de trading online îşi doresc un profit maxim, de aceea consideră preţul ca fiind cel mai important parametru QoS. Cu toate acestea, sunt dispuşi să ignore diferenţe mici de preţ (sub 10 cenţi) dacă serviciul compus având un preţ mai mare are valori mai bune ale timpului de execuţie şi fiabilităţii. Pentru clienţii sistemului este foarte important ca tranzacţiile să fie executate cât mai rapid. De aceea, sistemul de tranzacţii garantează un timp de execuţie sub 30 de secunde pentru cumpărarea unui fond. Pentru fiecare încălcare a

acestei garanţii, proprietarii sistemului de trading trebuie sa plătească o taxă de penalitate a cărei valoare depinde de durata întârzierii. Astfel pentru a compara două servicii web compuse, timpul de execuţie devine un criteriu important în cazul în care unul dintre serviciilor compuse are un timp de execuţie care depaăşeşte 30 de secunde. Metodele tradiţionale de ordonare bazate pe sumă ponderată sau pe asocierea de priorităţi parametrilor nu sunt potrivite pentru un astfel de scenariu. În continuare vom adresa mai multe probleme care apar în exemplul prezentat.

	$\succ 1$	$\succ 2$	$\succ 3$	$\succ 4$	$\sim$
WS1	1	0	2	0	1
WS2	1	0	1	0	0
WS3	0	0	0	0	0
WS4	1	0	1	0	0
WS5	1	1	0	0	0

4.2 Implementarea Binding-as-a-Service

În prezent există un număr mare de frameworkuri pentru compunerea serviciilor web, bazate pe diferite arhitecturi şi metodologii. Toate aceste frameworkuri au nevoie de funcţionalitatea de binding pentru a aloca servicii concrete activităţilor din cadrul modelului de orchestrare. De aceea se impune oferirea funcţionalităţii de „binding" sub forma unui serviciu web. Clientul tipic al serviciului de binding este un modul al unui framework de compunere de servicii care trebuie să selecteze serviciile optime pentru activităţile din modelului de compoziţie.

Noi oferim o implementare a funcţionalităţii BaaS bazată pe metoda de agregare a lui Yang et al. [3] şi pe metoda noastră de ordonare bazată pe preferinţe QoSPref. Prototipul BaaS este o aplicaţie Java şi este oferit ca proiect open-source la adresa: http://baas.sourceforge.net/.

O cerere a unui serviciu web client către aplicaţia server BaaS trebuie să conţină următoarele informaţii:

1. modelul de orchestrare;
2. lista de atribute QoS;
3. pentru fiecare activitate din cadrul modelului de orchestrare, o listă de servicii concrete care oferă funcţionalitatea cerută;
4. constrângerile QoS
5. preferinţele QoS.

Modelul de orchestrare este reprezentat de un workflow care are ataşat nodurilor sale probabilităţile de execuţie. În cazul în care probabilităţile de execuţie lipsesc, implementarea noastră va ataşa valori standard, după cum urmează: nodurilor care urmează după o poartă de tip XOR li se va ataşa probabilitatea 1/k, unde k este numărul de drumuri pornind din poarta XOR respectivă. Tuturor celorlalte noduri li se ataşează probabilitatea de execuţie 1.

Pentru specificarea workflourilor poate fi folosit direct formatul BPMN sau un format XML simplu, definit de noi.

Lista atributelor QoS trebuie să conţină informaţii despre categoria de agregare a fiecărui tip de atribut.

Lista serviciilor candidat pentru fiecare activitate trebuie să ofere şi informaţii despre valorile QoS ale fiecărui serviciu. Există situaţii în care nu toate serviciile web posedă toate atributele QoS specificate pentru modelul de compunere. Spre exemplu, un serviciu web compus care oferă funcţionalităţi grafice pot avea atribute QoS care precizează calitatea imaginilor oferite, spre exemplu numărul de culori, sau rezoluţia oferită. Serviciul compus poate avea un serviciu component care oferă funcţionalităţi de sortare, acesta neavând atribute QoS legate de numărul de culori şi rezoluţia imaginilor oferite de serviciul compus. În situaţia în care un atribut QoS lipseşte, implementarea noastră oferă valori standard, în funcţie de categoria de agregare a atributului QoS care nu este specificat.

4.3 Alegerea serviciilor web concrete folosind un algoritm genetic

Optimizarea valorii QoS agregate a unui serviciu compus este o problemă NP-hard. O căutare exhaustivă nu este posibilă decât pentru modele de compoziţie simple cu un număr mic de task-uri şi un număr mic de servicii disponibile pentru fiecare task. Implementarea noastră BaaS foloseşte un algoritm genetic pentru a găsi combinaţia optimă de servicii concrete care implementează modelul de orchestrare abstract. Acest algoritm va fi descris în paragrafele următoare.

Un algoritm genetic menţine o populaţie de cromozomi, în care fiecare cromozom codifică o posibilă soluţie a problemei. În cazul nostru, un cromozom codifică o posibilă mapare a serviciilor web la task-uri din modelul de orchestrare, aşa cum se prezintă în Fig. 9.

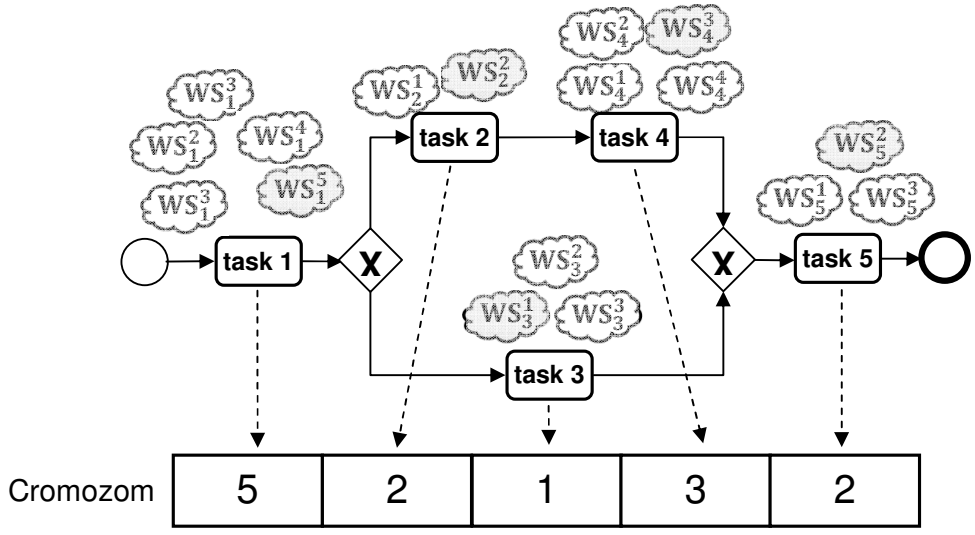

Fig. 9. Structura unui cromozom care codifică o posibilă mapare a serviciilor web la task-uri din modelul de orchestrare

Cromozomul este structurat ca un vector cu n elemente, unde n este numărul de task-uri din modelul de orchestrare. Valoarea elementului i din acest vector este un număr întreg care reprezintă indicele serviciului component asignat task-ului i. De exemplu, în Fig. 9 există 3 servicii care implementează funcţionalitatea task-ului cu numărul 5: WS_5^1, WS_5^2 şi WS_5^3. Soluţia codificată de cromozom mapează cel de-al doilea serviciu candidat (WS_5^2) la task-ul cu numărul 5. De aceea, valoarea celui de-al 5-lea element al cromozomului este 2.

Pentru recombinare, algoritmul nostru genetic foloseşte operatorul *two-point crossover*. Mutaţiile sunt realizate alegând în mod aleator un task şi schimbând indicele serviciului component asociat acestui task.

O particularitate a abordării noastre pentru specificarea preferinţelor este faptul că fitness-ul unei soluţii poate fi evaluat numai în contextul unei anumite populaţii, deoarece algoritmul de ranking (QoSPref) trebuie să execute comparaţii ale tuturor perechilor de soluţii candidat existente. Din acest motiv, nu este posibilă calcularea unei valori absolute a fitness-ului unei soluţii. Algoritmul nostru genetic calculează fitness-ul unei soluţii pe baza indicelui acesteia în lista de ranking obţinută de algoritmul QoSPref pentru populaţia curentă: soluţia din fruntea listei va avea fitness-ul maxim, iar cea de pe ultima poziţie fitness-ul minim. Pseudocodul procedurii de evaluare a fitness-ului este prezentat în Fig. 10.

```
procedure evaluateFitness()
        for i ← 1 .. n do
                qos[i] ← computeAggregateQoS(solution[i])
        end for
        for i ← 1 .. n-1 do
                for j ← i+1 .. n do
                        compResult[i][j] ← compareServices(solution[i], solution[j])
                end for
        end for
        createScoreVectors()
        rankList ← list of solutions sorted using compareScores()
        for i ← 1 .. n do
                fitness[i] ← populationSize - (index of solution[i] in rankList)
        end for
end procedure
```

Fig. 10. Procedura pentru evaluarea fitness-ului soluţiilor candidat

Funcţia *computeAggregateQoS* din pseudocodul de mai sus foloseşte metoda descrisă în secţiunea 2.4.3 pentru a estima costul serviciului compus pe baza valorilor QoS ale serviciilor componente. Funcţiile *compareSevices*, *createScoreVectors* şi *compareScores* au fost definite în secţiunea 3.2. Variabila *populationSize* este un parametru configurabil al algoritmului. În secţiunea 4.3.1 sunt prezentate rezultatele experimentale obţinute pentru diferite valori ale lui *populationSize*.

Există o serie de condiţii care pot fi combinate pentru a cauza terminarea algoritmului nostru genetic:

1. atingerea unui număr maxim de generaţii;
2. atingerea unei limite maxime a timpului de execuţie;
3. incapacitatea de a îmbunătăţi soluţia curentă în timpul ultimelor k generaţii.

Întrucât nu există o valoare absolută a fitness-ului unei soluţii, este dificil de verificat apariţia celei de-a 3-a condiţii. Pentru a rezolva această problema, algoritmul nostru genetic menţine o listă a celor mai bune soluţii găsite pâna la momentul curent. La sfârşitul fiecărei generaţii, cea mai bună soluţie din generaţia curentă este căutată în lista celor mai bune soluţii. Dacă nu este deja prezentă, ea este adăugată la această listă şi se calculează rangul ei folosind algoritmul QoSPref. Dacă noua soluţie este pe primul loc, înseamnă că am găsit o soluţie îmbunătăţită. Dimensiunea listei celor mai bune soluţii este limitată de o valoare configurată ca parametru al algoritmului. Dacă în urma adăugării noii soluţii dimensiunea listei depăşeşte această valoare limită, elementul cu cel mai slab rang va fi eliminat din listă.

4.3.1 Rezultate experimentale

În cadrul experimentelor noastre, am folosit 21 de modele de orchestrare din repositoriul public Oryx (disponibil la adresa https://code.google.com/p/oryx-editor/) şi 10 modele de orchestrare din biblioteca de procese business IBM BIT (disponibilă la adresa http://www.zurich.ibm.com/csc/bit/downloads.html). Modelele considerate au între 5 şi 20 de taskuri şi numărul de porţi variază între 0 şi 16. Am ales ca execuţia algoritmului să fie terminată atunci când după 20 de generaţii consecutive nu s-a obţinut nici o îmbunătăţire. Stabilim că dimensiunea maximă a listei cu cele mai bune soluţii este 10.

Am efectuat experimente folosind populaţii de dimensiunea 20, 50 şi 100. Pentru fiecare dimensiune a populaţiei şi pentru fiecare model de orchestrare am rulat 100 de iteraţii folosind un set diferit de servicii web candidat pentru fiecare iteraţie. Fiecare set de servicii candidat a fost creat generând între 2 şi 7 alternative pentru fiecare activitate din cadrul modelului de orchestrare curent.

Rezultatele experimentale au dovedit că şi o populaţie de 20 de invidivizi este suficientă pentru a atinge condiţia de terminare a rulării în mai puţin de 50 de generaţii. În timp ce, de obicei, folosirea unui populaţii mari conduce la un număr mic de generaţii, diferenţa nu este considerabilă faţă de folosirea de populaţii mici, iar, pentru anumite modele de orchestrare, o populaţie mare poate conduce chiar la rezultate mai proaste.

De asemenea, rezultatele experimtelor arată că numărul de generaţii necesare pentru atingerea condiţiei de terminare a rulării algoritmului este dependent de numărul de activităţi şi porţi, dar acest număr mai este influenţat şi de alţi factori, cum ar fi topologia modelului de orchestrare.

5. Extensia ontologiei OWL-Q pentru specificarea preferinţelor

În acest capitol, extindem ontologia OWL-Q pentru a putea folosi în cadrul descoperii semantice flexibilitatea metodei de exprimare a cerinţelor QoS prezentate în secţiunea anterioară.

5.1 Specificarea cerinţelor QoS în cadrul ontologiei OWL-Q

În cadrul OWL-Q cererile şi oferta atributelor QoS sunt definite de faţeta QoSSpec. Clasa QoSSpec conţine descrierea QoS a serviciului web. Această faţetă conţine diferite atribute QoS de bază cum ar fi costul serviciului şi moneda în care este exprimat, atributele de securitate şi protocoalele acceptate, perioada de valabilitate a ofertei. QoSSpec cuprinde doua clase separate QoSOffer şi QoSDemand care sunt folosite de furnizorii de servicii web, respectiv de clienţii serviciilor web pentru a specifica folosind aceeasi metodă, într-un mod simetric, constrângerile QoS.

Clienţii serviciilor web pot specifica constrângerile QoS folosind clasa QoSDemand şi pot asocia ponderi metricilor căutate folosind clasa QoSSelection. Clasa QoSSelection conţine o lista de intrări <metrică, pondere>. Ponderea asociată poate avea valorea 2.0 dacă este o constrângere obligatorie sau o valoare aflată în intervalul (0:0; 1:0) dacă este o preferinţă. O intrare <metrică, pondere> este definită de clasa QosSelectElem, iar clasa QoSSelectElemeList conţine o listă de elemente QosSelectElem. Pentru a ilustra mai bine clasele folosite pentru specificarea cerinţelor QoS şi a relaţiilor dintre ele, Fig. 11 prezintă graful care reprezintă faţeta QosSpec. Acest graf a fost generat folosind editorul Protégé.

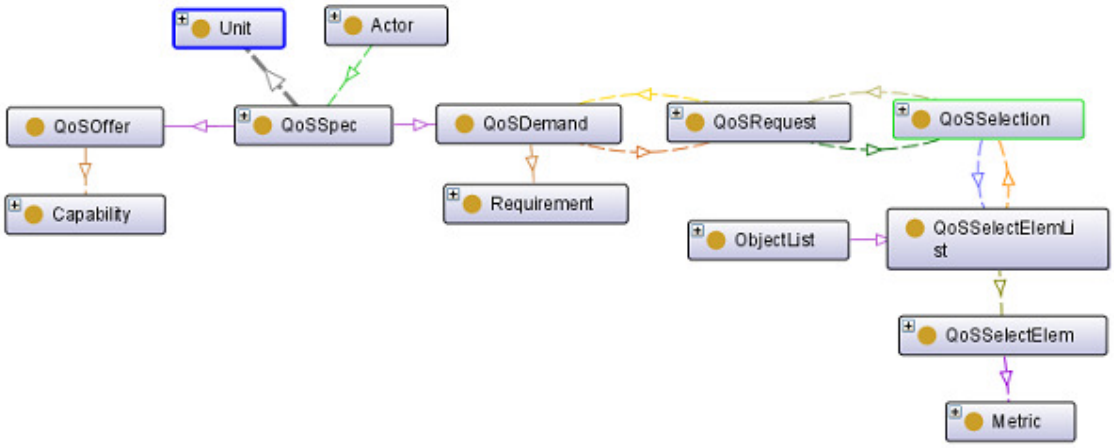

Fig. 11. Faţeta QoSSpec a ontologiei OWL-Q

OWL-Q conţine şi o faţetă care descrie tipurile de valori pe care un atribut QoS le poate avea. Clasa MetricValueType este împărţită în două subclase, care indică direcţia valorilor atributului QoS. Tipul PositivelyMonotonic indică faptul că valorile sunt ordonate de la cea mai scazută către cea mai mare valoare iar valoarea maximă este considerată ca fiind cea mai bună valoare. Tipul de valoare NegativelyMonotonic are o direcţie contrară, valoarea maximă de calitate fiind asociată valorii minime. Pentru a exemplifica, atributul QoS care măsoară disponibilitatea unui sistem este o valoare de tipul PositivelyMonotonic, în timp ce timpul de răspuns sau costul unui serviciu sunt valori de tipul NegativelyMonotonic deoarece valorile mici indică o calitate mai bună. Astfel, atunci când specificăm o preferinţă folosind metoda propusă de noi QosPref, nu avem nevoie sa specificăm explicit direcţia valorilor optimale pentru atributul respectiv acest lucru fiind deja specificat folosind această faţetă.

5.2 Extensia OWL-Q propusă

Pentru descrierea calităţii ofertei de servicii realizate de către furnizorii acestora, vom folosi în continuare clasa QoSOffer din cadrul OWL-Q. Noi vom extindem faţeta QoSSpec adăugând noi clase pentru selecţia preferinţelor.

Definim o noua clasă QoSSelectionWithTradeoffs care va permite specificarea unei liste de preferinţe, analog notaţiei oferite de metoda QoSPref. Constângerile sunt specificate folosind clasa QoSDemand, la fel ca în OWL-Q.

O preferinţă este descrisă de clasa QoSPreference care defineşte formatul unei reguli din blocul de preferinţe. În cadrul metodei QosPref, o regulă din blocul de preferinţe conţine trei componente: o condiţie opţională, atributul QoS care stă la baza comparării şi direcţia valorilor optimale pentru atributul respectiv. Deoarece pentru specificarea direcţiei valorilor optimale vom folosi clasa MetricValueType, o regulă din cadrul blocului de preferinţe va conţine doar condiţia opţională şi atributul comparat. O intrare din cadrul blocului de preferinţe are următorul format: <(condiţie opţională) atriburt QoS>.

Pentru specificarea condiţiei opţionale definim trei operatori logici: AT_LEAST_ONE (operator OR), EXACTLY_ONE (operator XOR), ALL (operator AND) şi operatorul aritmetic DIFF (-), aşa cum sunt definiţi în Fig. 2, pe care îi includem în cadrul ontlogiei. Condiţia opţională este specificată sub forma <operator, atribut >.

În cadrul metodei QoSPref ordinea preferinţelor este importantă pentru algoritmul de sortare, prima preferinţă specificată fiind considerată ca fiind cea mai important. Astfel, ordinea în care sunt definite preferinţele din cadrul liste de preferinţe este importantă. Pentru ca notaţia semantică să fie explicită şi uşor de urmărit ataşăm în mod explicit prioritatea preferinţei fiecărei intrări din lista de preferinţe. Astfel clasa QoSPreference este definită de următoarea notaţie: <prioriatate, (condiţie) atribut>.

Extensia ontologiei este expusă în Fig. 12.

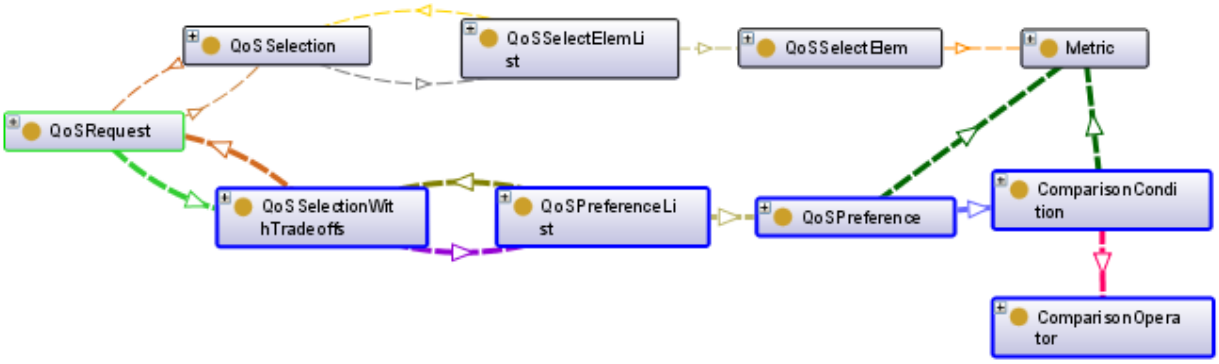

Fig. 12. OWL-Q extension compatible with the QoSPref approach

Clasa QoSRequest va putea folosi noua clasă QoSSelectionWithTradeoffs sau formatul vechi de specificare a preferinţelor QoSSelection. Clientul poate decide ce clasă pentru specificarea preferinţelor va folosi, în funcţie de complexitatea cerinţelor sale.

Extensia oferită de noi îi permite clientului sa folosească o notaţie puternică pentru a specifica cerinţe QoS complexe, într-un cadru semantic.

6. Concluzii

În cadrul tezei de doctorat am introdus o nouă abordare inovatoare pentru compunerea serviciilor web, ţinând cont de calitatea serviciilor (QoS). Abordarea propusă oferă un cadru complet, care adresează toate etapele necesare realizării unei compuneri de servicii web, plecând de la specificarea cerinţelor clientului, descoperirea serviciilor web într-un mod semantic bazat pe cerinţele clientului, specificarea modelului de compunere şi găsirea serviciilor concrete optime care împreună să realizeze serviciul compus optim pentru cerinţele exprimate .

În prezent, arhitecturile bazate pe servicii beneficiază de un interes major. Oferta de servicii web este foarte mare, iar problema compunerii dinamice a serviciilor web ţinând cont de calitatea acestora este o direcţie de cercetare actuală de mare interes. Cu toate acestea, nu există în momentul de faţă un standard care să se fi impus.

În cadrul tezei de doctorat analizăm cele mai importante abordări propuse şi identificăm problemele acestora. Soluţia propusă se bazează pe observaţia că posibilitatea exprimării de către clienţi a unor cerinţe de calitate QoS complexe, care să permită specificarea atât a cerinţelor cât şi a preferinţelor de calitate şi a unor reguli de compromis, este de o importanţă crucială pentru găsirea soluţiei optime pentru fiecare client în parte.

Am oferit o metodă de specificare a preferinţelor, bazată pe un limbaj intuitiv şi simplu de utilizat, dar în acelaşi timp expresiv, numit QoSPref, care permite definirea unor cerinţe de calitate complexe. Am observat că descoperirea serviciilor web necesare, bazată pe o căutare sintactică oferă dezavantaje şi am ales să folosim o căutare semantică bazată atât pe criterii funcţionale cât şi de calitate. În acest scop am analizat ontologiile QoS existente şi am extins ontologia OWL-Q, care permite specificarea simetrică a capacităţilor de calitate ale serviciilor şi cerinţele de calitate ale clientului. Extensia propusă permite specificarea preferinţelor şi a regulilor de compromis folosind limbajul QoSPref. Astfel, metoda de specificare a preferinţelor clienţilor este integrată şi în cadrul procesului de selecţie semantic.

Pentru a ordona serviciile în funcţie de preferinţele specificate de client, am propus un algoritm simplu, care nu se bazează pe tehnici multicriteriale complexe şi care realizează o ordonare totală a alternativelor. Algoritmul poate face faţă unor preferinţe intranzitive. În acest moment, algoritmul realizează o optimizare locală a etapei de căutare a serviciilor web.

Am oferit motorul de selecţie sub forma unei biblioteci Java, pe care am pus-o la dispoziţie sub forma unui proiect open-source denumit QoSPref, accesibil la adresa http://qospref.sourceforge.net. Un scenariu tipic de ultilizare a acestei biblioteci este ca parte integrantă a unui framework SOA, care accesează motorul de selecţie prin intermediul interfeţei de programare (API) puse la dispoziţie de biblioteca noastră. Framework-ul SOA, care joacă rol de client al motorului de selecţie, trebuie să ofere ca date de intrare informaţii legate de serviciile web candidat, de atributele QoS ale acestora, precum şi de preferinţele QoS care trebuie luate în considerare.

În plus, pentru a ilustra modul de utilizare a bibliotecii şi a uşura înţelegerea funcţionării motorului de selecţie, QoSPref oferă şi o interfaţă grafică prin intermediul căreia pot fi accesate funcţiile oferite de bibliotecă.

Pentru a realiza o optimizare globală a compunerii de servicii, calculăm valorile agregate ale atributelor QoS, folosind funcţiile de agregare definite în cadul lucrării [**17**]. Algoritmul anterior de ordonare a alternativelor este folosit de această dată pentru a compara valorile de calitate agregate pentru diferitele posibile soluţii de compunere.

Pentru alegerea serviciilor concrete care implementează activităţile existente într-un model de compunere am oferit, sub forma unui serviciu web, metoda BaaA (Binding-as-a-Service). Această metodă aduce o îmbunătăţile majoră faţa de metodele existente, deoarece permite ca workflowuri de intrare atât workflowuri structurate, cât şi nestructurate, fiind astfel foarte flexibilă şi depăşind constrângerile celorlalte abordări existente pentru această problemă. BaaS necesită ca parametri de intrare: specificaţia workflowului abstract, specificaţie ce poate fi realizată în BPMN, în formatul proprietar IBM Business Modeller sau într-un format simplu, propriu metodei noastre, lista de cerinţe QoS a clientului şi lista serviciilor web concrete care implementează fiecare activitate din cadul workflowului abstract. Pentru calculul valorilor QoS agregate am folosit metoda eficientă de calcul introdusă în [**5**]. Această metodă calculează valoarile QoS agregate pe baza unor modele de orchestrare care pot conţine şi componente nestructurate.

Implementarea funcţionalităţii BaaS bazată pe metoda de agregare a lui Yang et al. [3] şi pe metoda noastră de ordonare bazată pe preferinţe QoSPref este oferită proiect open-source la adresa: http://baas.sourceforge.net/. Proiectul este o aplicaţie Java care oferă funcţionalitatea implementată sub forma unui serviciu web. Clientul tipic al serviciului de binding este un modul al unui framework de compunere de servicii care trebuie să selecteze serviciile optime pentru activităţile din modelului de compoziţie.

Deoarece optimizarea valorii QoS agregate a unui serviciu compus este o problemă NP-hard, o căutare exhaustivă a serviciilor concrete optime nu este posibilă decât pentru modele de compoziţie simple cu un număr mic de task-uri şi un număr mic de servicii disponibile pentru fiecare task. Pentru a găsi combinaţia optimă de servicii concrete care implementează modelul de orchestrare abstract amfolosut un algoritm genetic. Algoritmul genetic a fost modificat pentru a folosi, ca funcţie de fitness algoritmul de ordonare propus de noi.

Rezultatele experimentale, bazate pe workflowuri de intrare specificate în BPMN din cadrul repositourului academic public Oryx şi a unor workflowuri din cadrul repositourului public BIT, specificate în limbajul proprietar al IBM Business Modeller demonstrază eficacicatea metodei propuse.

Referințe bibliografice

[1] Raluca Iordache and Florica Moldoveanu, "A conditional lexicographic approach for the elicitation of QoS Preferences," in *Proceedings of the 20th International Conference on Cooperative Information Systems (CoopIS 2012)*, Rome, 2012.

[2] Kyriakos Kritikos and Dimitris Plexousakis, "OWL-Q for Semantic QoS-based Web Service Description and Discovery," *Proceedings of the SMR2 2007 Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web*, pp. 123-137, 2007.

[3] Raluca Iordache and Florica Moldoveanu, "QoS-aware web service semantic selection based on preferences," in *24th DAAAM International Symposium on Intelligent Manufacturing and Automation*, Zadar, Croatia, 2013.

[4] Raluca Iordache and Florica Moldoveanu, "A web service composition approach based on QoS preferences," in *Proceedings of the 6th IEEE International Conference on Service Oriented Computing & Applications (SOCA 2013)*. Kauai, Hawaii, 2013, pp. 220-224.

[5] Y. Yang, M. Dumas, L. Garcia-Banuelos, A. Polyvyanyy, and L. Zhang, "Generalized aggregate Quality of Service computation for composite services," *Journal of Systems and Software*, no. 85(8), pp. 1818 - 1830, 2012.

[6] Raluca Iordache and Florica Moldoveanu, "A genetic algorithm for automated service binding," in *24th DAAAM International Symposium on Intelligent Manufacturing and Automation*, Zadar, Croatia, 2013.

[7] Raluca Iordache, Serban Iordache, and Florica Moldoveanu, "A Framework for the Study of Preference Incorporation in Multiobjective Evolutionary Algorithms," in *Proceedings of the 16th Annual Conference on Genetic and Evolutionary Computation, GECCO 2014*. Vancouver, Canada: ACM, 2014, p. (to appear).

[8] Thomas Allweyer, *BPMN Business Process Modeling Notation*.: Books on Demand.

[9] ISO, "ISO 8402:1994 - Quality management and quality assurance - Vocabulary," 2004.

[10] I. Burnstein, *Practical Software Testing: A Process-Oriented Approach*.: Springer, 2003.

[11] Marc Oriol Hilari, "Quality of Service (QoS) in SOA Systems A Systematic Review," Universitat Politècnica de Catalunya, 2009.

[12] Jorge Cardoso, *Quality of Service and Semantic Composition*. USA, University of Georgia, 2002.

[13] J.A. Miller, J.S. Cardoso, and G. Silver, "Using Simulation to Facilitate Effective Workflow Adaptation," in *Proceedings of the 35th Annual Simulation Symposium (ANSS'02)*, San Diego, 2002, pp. 177-181.

[14] Mike P-Papazoglou,Willem-Jan van den Heuvel Jian Yang, "Tackling the challenges of service composition in e-marketplaces," Proceedings of the 12th International Workshop on Research Issues in Data Engineering (RIDE '02) 2002.

[15] Gero Mühl, Michael C. Jaeger Gregor Goldmann, "QoS Aggregation for Web Service Composition using Workflow Patterns," Berlin University of Technology, Germany,.

[16] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani, "An Approach for QoS-aware Service Composition based on Genetic Algorithms," RCOST - Research Centre on Software Technology University of Sannio , Italy,.

[17] J. Cardoso, A. Sheth, J. Miller, J Arnold, and K Kochut, "Quality of service for workflows and web service processes," in *Web Semantics: Science, Services and Agents on the World Wide Web 1.*, 2004, pp. 281-308.

[18] A. Polyvyanyy, L. Garcia-Banuelos, and M. Dumas, "Structuring Acyclic Process Models," in *Proceedings of the International Conference on Business Process Management pp. 276–293*, New York, USA, 2010.

[19] R.T. Marler and J.S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and Multidisciplinary Optimization*, 2010.

[20] Eckart Zitzler, "Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications," Institut für Technische Informatik Eidgenössische Technische Hochschule Zürich, Zürich, PHD.

[21] J. El Haddad, M. Manouvrier, and M. Rukoz, "TQoS: Transactional and QoS-Aware Selection Algorithm for Automatic Web Service Composition," *IEEE Transactions on Services Computing*, vol. 3, no. 1, pp. 73-85, 2010.

[22] Florian Rosenberg, Philipp Leitner, Anton Michlmayr, Predrag Celikovic, and Schahram Dustdar, "Towards Composition as a Service – A Quality of Service Driven Approach," Distributed Systems Group, Technical University Vienna, Vienna,.

[23] L. Zeng et al., "QoS-Aware Middleware for Web Services Composition," in *IEEE Trans. on Software Engineering, 30, issue 5, pp. 311-327*, 2004.