Scientific Journal of Riga Technical University
Computer Science. Technologies of Computer Control
2010
Volume 42

Biznesa procesu simulāciju un transformāciju izmantošana elektronisko pakalpojumu projektēšanā

Peteris Stipravietis, *Riga Technical University*

***Kopsavilkums*. Rakstā tiek aplūkota biznesa procesu izstrādes problēmu risināšana ar darbplūsmu valodas YAWL palīdzību. Autora piedāvātā pieeja balstīta uz biznesa procesu definēšanu YAWL vidē, lai procesa modelim pēc tam varētu veiktu simulācijas, ar kuru palīdzību varētu atrisināt aprakstītās projektēšanas problēmas un uzlabot sākotnējo modeli. Tāpat rakstā parādīts, kā no uzlabotā YAWL modeļa iegūt procesa primitīvo aprakstu kā orientētu grafu, kuru pēc tam izmanto, lai YAWL darbplūsmu pārveidotu uz procesa aprakstu citā, hierarhiskā valodā. Procesa primitīvajā aprakstā tiek identificēti procesu šabloni, no kuriem pēc tam tiek izveidots biznesa procesa apraksts BPEL. Iegūtais rezultāts tiek novērtēts, salīdzinot ar sākotnējo darbplūsmu un balstoties uz iespējām jaunizveidoto procesu darbināt kādā BPEL procesu izpildes vidē, pieliekot iespējami mazāk programmētāja darbu.**

***Atslēgas vārdi*: YAWL, BPEL, transformācija, simulācija.**

I. Ievads

E-pakalpojumi ieņem arvien lielāku vietu mūsdienu informācijas sabiedrībā, tie kļūst arvien pieejamāki un daudzveidīgāki, taču problēmas, ar kurām jāsastopas to projektēšanas un izstrādes laikā, pārsvarā ir vienas un tās pašas – kā ērtāk aprakstīt biznesa procesu lietotājam bez īpašām iemaņām programmēšanā, kā uzskatāmāk un vienlaikus precīzāk definēt procesu, nezaudējot par to svarīgu informāciju, kā arī – kā pārbaudīt izveidoto biznesa procesa modeli un identificēt iespējamās problēmas jau projektēšanas fāzē, pirms tās tiek implementētas – noskaidrot tā 'vājās' vietas pirms darbināšanas, veikt procesa skaņošanu, balstoties uz mērījumiem u.c. Pārbaude un procesa validācija ļoti svarīga ir arī tad, ja e-pakalpojuma process tiek mainīts jau pēc tā ieviešanas – jāpārliecinās, vai izmaiņas ir korekti implementētas, vai vienlaicīgi spēj izpildīties gan jaunā procesa instances, gan tās vecā procesa instances, kuras vēl nav pabeigtas. Šo izmaiņu veikšanai un pārbaudei jābūt arī pietiekami ērtai un konkurētspējīgai izmaksu ziņā – līdz ar to jāsecina, ka šādu grūtību risināšanai liela nozīme ir izvēlētajai valodai, ar kuru tiks aprakstīts biznesa process – vai tā satur pietiekami daudz izteiksmes līdzekļu, vai ar to ir iespējams validēt procesu modeļus, vai ir iespēja izstrādātos modeļus simulēt.

Esošās biznesa procesu modelēšanas valodas var iedalīt divās lielās grupās. Pirmās grupas valodas ir iecienītas akadēmiskajās aprindās, taču tiek maz izmantotas reālos risinājumos. Šo valodu pamatā ir Petri tīkli, procesu algebra, tām ir atbilstoša formālā semantika, kas ļauj modeļiem, aprakstītiem ar šīm valodām, tikt formāli validētiem. Otrās grupas valodas biežāk tiek izmantotas reālos risinājumos, nevis akadēmiskos pētījumos – tās ir BPEL [17], WSFL [19] un citas. Šīm, tā saucamajām biznesa valodām, bieži trūkst atbilstošas semantikas, kas noved pie debatēm, kā pareizi interpretēt biznesa modeļus. Fakts, ka šo valodu procesu izpildītājiem arī ir pieejamas atšķirīgas, vairāku izstrādātāju izveidotas implementācijas, situāciju nepadara vieglāku. Neskatoties uz to, praksē vairāk tiek izmantotas tieši biznesa valodas, savukārt akadēmiskie modeļi tiek izmantoti samērā reti. Saskaroties ar situāciju, kad biznesa modelim jāveic verifikācija, teiksim, balstīta uz Petri tīkliem, ir jāatsakās no šādas biznesa procesa pārbaudes, vai arī process jātransformē uz citu procesu, kas aprakstīts uz Petri tīkliem bāzētā valodā, piemēram, YAWL [1]. Autors piedāvā procesa izstrādi veikt otrā virzienā – vispirms tā modeli izstrādāt akadēmiskā valodā, modeļa nepilnību identificēšanai veikt analīzi ar matemātiskiem līdzekļiem un pēc tam to transformēt uz procesu, kas aprakstīts kādā no biznesa valodām. Šādas pieejas galvenās priekšrocības ir sekojošas:

- Modeļi, kas sākotnēji aprakstīts ar akadēmisku valodu, ir vieglāk uztvert, uzturēt, kā arī veikt tā analīzi, salīdzinot ar tādu, kas ir konvertācijas rezultāts no kādā citā valodā aprakstīta modeļa. Konvertēts modelis turklāt konvertācijas gaitā var būt ticis būtiski pārveidots un zaudējis daļu no projektējuma informācijas.
- Modelis, kas transformēts uz biznesa valodu, ir jau validēts un to var sākt darbināt. Protams, pirms tam modelis būtu jāpārskata un jāveic kādi labojumi, kurus transformācija no akadēmiskās valodas modeļa nav spējusi apstrādāt. Modelim, kas aprakstīts ar biznesa valodu, daudz vienkāršāk piemeklēt izpildes vidi, kurai pie tam ir nodrošināts tehniskais atbalsts.

Šī raksta mērķis ir pārliecināties, vai piedāvātā pieeja ir pielietojama vienkārša biznesa procesa projektēšanā, uzvaru liekot uz pārbaudi, vai ar tās palīdzību iespējams identificēt biežāk sastopamās procesu nepilnības, kā arī aplūkot transformācijas no akadēmiskās valodas YAWL uz biznesa valodu BPEL – šīs valodas ir izvēlētas minētās pieejas realizācijai.

II. Simulācija

Lielu nozīmi biznesa procesa projektēšanā ieņem uzprojektētā procesa analīze – šauro vietu atrašana, kurās viena procesa instance vai tās fragments var aizņemt visus pieejamos resursus, līdz ar to liekot citām instancēm gaidīt uz to atbrīvošanos; strupceļu identificēšana – aktivitātes biznesa procesā, kas var novest pie procesa 'ieciklēšanās' vai mūžīgas gaidīšanas, neļaujot procesa beigties; klinču (strupsaķeru – deadlock) atrašana – situācija, kad vairāki procesi, kas

pieprasa vienus un tos pašus resursus, bloķē viens otru; kļūdu apstrādes mehānismi, kad jāatceļ iepriekšējās aktivitātēs veiktās darbības; atkalizmantojamo procesa bloku, piemēram, audita datu rakstīšanas sistēmas žurnālā, identificēšana.

Šādus uzdevumus iespējams risināt ar simulācijas palīdzību. Tālāk aplūkosim, kā veikt YAWL darbplūsmu simulāciju, kas ļaus noskaidrot, kurus no minētajiem uzdevumiem iespējams risināt ar simulācijas palīdzību. Autors izmanto darba [6] autoru piedāvāto simulācijas paņēmienu, kurā tiek izmantota gan procesa projektējuma dati, gan vēsturiskie dati par procesa instancēm no audita pierakstiem, kā arī procesa instanču stāvokļu dati no procesa izpildāmās vides un darbplūsmas organizācijas modelis. No minētajiem informācijas avotiem tiek izveidots simulācijas modelis – projektējuma dati tiek izmantoti, lai aprakstītu simulācijas modeļa struktūru; vēsturiskie dati nosaka simulācijas modeļa darbināšanas parametrus; stāvokļu dati tiek izmantoti, lai inicializētu simulācijas modeli, savukārt organizācijas modeļa dati apraksta aktivitātēm pieejamos resursus.

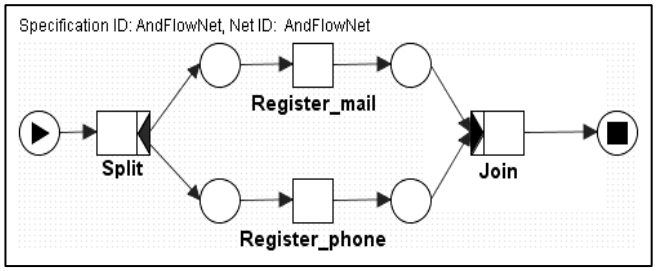
Pārveidojot konvertēto simulācijas modeli, iespējams modelēt dažādas situācijas – piemēram, izlaist kādu darbību, novirzīt procesa plūsmu pa citu zaru. Ņemot vērā esošo procesa instanču stāvokļu datus, iespējams aplūkot sistēmas stāvokli tuvā nākotnē (short-term simulation) un iegūto informāciju izmantot lēmumu pieņemšanā.

Darbplūsmas simulācija tiek veikta, izmantojot procesu datu izguves rīku ProM [7]. Lai izveidotu simulācijas modeli, tiek veikti šādi soļi:

- No procesa izpildāmās vides tiek importēti darbplūsmas projektējuma dati, organizācijas modeļa dati, kā arī audita pieraksti;
- No šīs informācijas tiek izveidots augsta līmeņa YAWL darbplūsmas modelis, kuru papildina ar esošo procesa instanču stāvokļu datiem (kādā stāvoklī katra procesa instance pašlaik atrodas);
- Jaunizveidotais modelis tiek konvertēts uz Petri tīklu;
- Iegūtais Petri tīkls tiek eksportēts uz simulācijas izpildes vidi CPN Tools [8] kā krāsains Petri tīkls. Esošo instanču dati tiek izmantoti kā tīkla sākotnējie parametri, aprakstīti valodā ML.

Simulācijas izpildes vidē CPN Tools var veikt procesa darbības simulāciju gan ilgākā laika posmā, gan reālā laikā, izmantojot kādas konkrētas procesa instances stāvokļa informāciju. Šis paņēmiens no citiem autoram zināmajiem atšķiras tieši simulācijas reālisma ziņā – daudzi citi simulācijas modeļi pieņem, ka pieejamie resursi tiek veltīti tikai simulējamai procesa instancei, neņemot vērā reālās situācijas, kad resursus izmanto vairākas instances, kā arī citi procesi, savukārt dotais paņēmiens simulācijas modelī izveido mākslīgas aiztures, balstoties uz vēsturiskajiem izpildes datiem no audita pierakstiem.

Atgriežoties pie projektēšanas laikā sastopamajām problēmām, turpmāk rakstā tiks novērtēts darbā [6] aprakstītais simulācijas paņēmiens – vai to var izmantot, lai minētās problēmas identificētu un spētu novērst pirms e-pakalpojuma biznesa procesa ieviešanas produkcijā.



1.att. YAWL darbplūsma – paralēlā plūsma

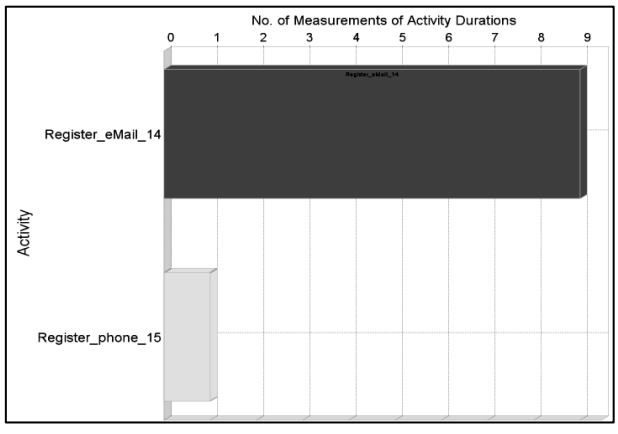
Samērā izplatīta kļūda ir pieņemt, ka pakalpojuma lietotājs ievadīs visu no viņa prasīto informāciju – daļu no tās viņš negribēs izpaust, daļas no tās viņam vispār nav. Ja process sagaida šādu informāciju, bet tā netiek norādīta, tad process principā nonāk strupceļā – gaida kādas aktivitātes beigas, kuras nekad nepienāks. Kā piemērs tiek parādīta vienkārša YAWL darbplūsma, kas no klienta sagaida e-pasta un tālruņa numura norādīšanu (sk. 1. attēlu). Plūsma ir realizēta ar AND plūsmu – abi zari ir obligāti jāizpilda, lai process beigtos.

Šādu problēmu risina, AND plūsmu aizstājot ar OR plūsmu, taču aplūkotais simulācijas paņēmiens šajā situācijā īsti nav izmantojams – no YAWL darba vides izgūtajām, gaidošajās procesa instancēm tiek ģenerēts sintaktiski nepareizs SML fails, ja YAWL tīkla izejas mainīgajiem plūsmā vēl nav piešķirta vērtība, savukārt piešķirt tiem inicializācijas vērtības projektēšanas laikā nav iespējams.

Arī pēc tīkla inicializācijas faila labošanas simulācija neļaus atklāt problēmu, jo arī krāsainajā Petri tīklā tiek ģenerēta AND plūsma – īstermiņa simulācija veiks procesa izpildi pa abiem zariem. Labot tīklu nav jēgas, jo tajā brīdī problēmu jau būtu identificēta un nebūtu vajadzības to vēlreiz simulēt.

Lai arī izrādās, ka dotās problēmas risināšana ar simulāciju nav iespējama, tomēr to var identificēt, analizējot izpildīto un esošo plūsmu audita datus, izmantojot ProM analīzes rīkus, piemēram, 'Basic Log Analysis'.

2. attēlā skaidri redzama atšķirība starp izpildītajām aktivitātēm. Līdz ar to rodas jautājums, kāpēc tālrunis tiek reģistrēts mazāk un kāpēc ir tik daudz nepabeigtu procesu. Kā jau tika minēts, problēmu risina, AND plūsmu aizstājot ar OR plūsmu, savukārt Petri tīkliem trūkst OR plūsmas atbalsta un šāds process nav simulējams.

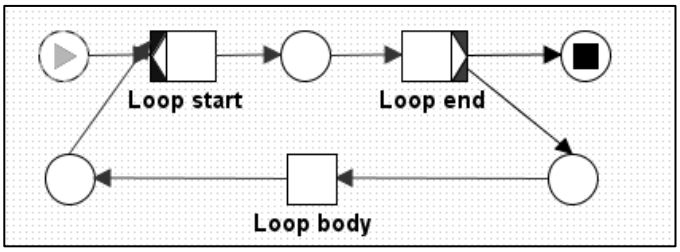


2.att. Izpildītās aktivitātes

Cita problēma ir procesa 'ieciklēšanās' – kādā no procesa cikliem piemirsts korekti nodefinēt izejas no cikla nosacījumus, vai arī cikla mainīgajam netiek mainīta vērtība, kā arī citi iemesli. Lai arī to bieži var konstatēt ar jau minēto ProM analīzes rīku 'Basic Log Analysis' (grafikā attiecīgās cikla aktivitātes būtu nedabīgā vairākumā), šī situācija tomēr ir atrodama arī ar simulācijas paņēmienu.

3. attēlā redzams cikla piemērs. Darbplūsmā ir definēts viens lokālais mainīgais – 'exitLoop', taču tā vērtība ar nolūku netiek mainīta. 4. attēlā redzams izveidotais Petri tīkls.

Uzskatāmības labad tam pievienoti mainīgie 'loopCount' un 'caseId' (iezīmes datu tips tiek definēts kā product INT*BOOL*INT). Veicot tīkla simulāciju, redzams, ka iezīmes nekad nepamet ciklu.

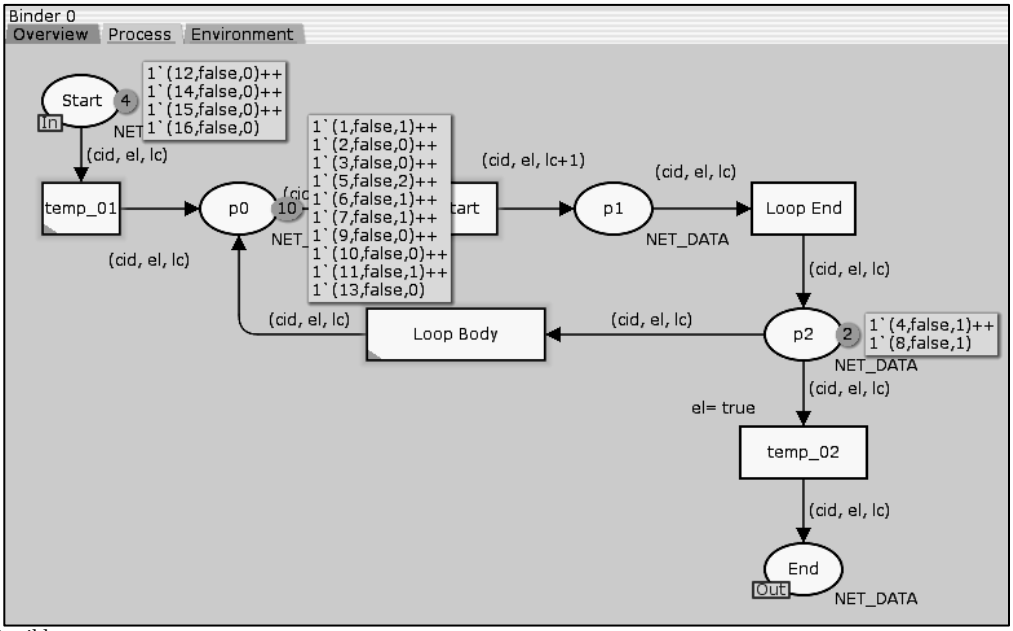


3.att. YAWL darbplūsma – bezgalīgais cikls

Līdzīga aina vērojama, cenšoties identificēt procesa šaurās vietas – arī tad šaurajās tīkla pārejās atrodas vairāk iezīmju kā citās. Iepriekš aplūkotajam piemēram par šauro vietu varētu uzskatīt cikla sākumu – tajā nonāk iezīmes gan no procesa sākuma, gan no paša cikla.

Veicot YAWL darbplūsmas simulāciju ar piedāvāto paņēmienu, autors secina, ka to iespējams pielietot, lai atrisinātu daļu no ievadā minētajām projektēšanas problēmām. Klinču atrašana diemžēl nav iespējama, jo aplūkotais simulācijas paņēmiens nepieļauj vairāku paralēlu procesa instanču simulāciju vienā apgabalā (scope).

Šis paņēmiens nenodrošina arī darbību atcelšanu, tāpēc nav iespējama YAWL kļūdu apstrādes mehānismu pārbaude. Šai problēmai cēlonis ir kļūdu apstrādes reģionu jēdziena neesamība Petri tīklos. Tāpat Petri tīkli neatbalsta OR plūsmas sadali (izpilde pa M no N zariem) un sinhronizāciju, līdz ar to arī šādu konstrukciju simulācija nav iespējama



4.att. Petri tīkls – bezgalīgais cikls

III. TRANSFORMĀCIJA

YAWL – Yet Another Workflow Language – darbplūsmu un biznesa procesu aprakstīšanas valoda, kas atbalsta sarežģītas datu transformācijas un tīmekļa servisu integrāciju. YAWL ir balstīta uz uzlabotajiem darbplūsmu tīkliem (EWF)[1]. YAWL var uztvert arī kā Petri tīklu, kas papildināts ar iespēju definēt vairākas paralēlas procesa instances, sinhronizācijām, OR konstrukcijām un atcelšanas mehānismiem. Darbplūsmas definīcija valodā YAWL ir hierarhiski sakārtota EWF tīklu kopa, kas veido koka struktūru.

Katrs tīkls sastāv no aktivitātēm un nosacījumiem – tīkla pozīcijām un pārejām. Aktivitātes ir vai nu atomāras, vai saliktas, katra saliktā aktivitāte patiesībā ir atsevišķs tīkls. Katram tīklam ir savs ieejas un izejas nosacījums [2]. YAWL ir savs vizuālais redaktors, ar kura palīdzību izveidot darbplūsmas, kā arī izpildes vide, kurā ielādēt pabeigtās un validētās darbplūsmu definīcijas. Ar YAWL izveidotie risinājumi tiek izmantoti biznesa vidē, piem. YAWL paplašinājums YAWL4Film tiek izmantots Austrālijas filmu, televīzijas un radio skolā (AFTRS) [3], bet paplašinājums YAWL4Health Akadēmiskajā medicīnas centrā (AMC) Nīderlandē [4]. Kā pētījuma akadēmiskā valoda YAWL izvēlēta šādu iemeslu dēļ:

- YAWL ir balstīta uz EWF, tās darbplūsmas iespējams transformēt uz krāsainiem Petri tīkliem un veikt to simulāciju, ka arī formālu semantisko validāciju;
- YAWL ir speciāli izstrādāta tā, ka tā atbalsta visus iespējamos darbplūsmu šablonus [5].

BPEL – Business Process Execution Language – biznesa procesu izpildāmā valoda. Procesi, kas aprakstīti ar BPEL, informācijas apmaiņai ar citiem procesiem vai sistēmām izmanto tīmekļa servisus. BPEL ir balstīta uz XML, tai nav noteikta grafiska pieraksta. Kā pētījuma biznesa valoda BPEL izvēlēta šādu iemeslu dēļ:

- BPEL ir industrijas standarts;
- BPEL ir balstīta uz tīmekļa pakalpojumu izmantošanu, kas to padara ļoti piemērotu tieši e-pakalpojumu projektēšanā un darbināšanā.
- Ir pieejamas vairākas populāras vides, kurās izstrādāt un darbināt BPEL procesus – Microsoft BizTalk, Oracle BPEL Process Manager, IBM WebSphere u.c.

Ar BPEL procesu transformāciju uz YAWL darbplūsmām nodarbojušies A. Brogi un R. Popescu [9]. Šī darba pamatā ir katra BPEL elementa transformācija uz YAWL elementu šabloniem. Kā galvenais iemesls šādas transformācijas izstrādei tiek minēta esošo BPEL formalizācijas paņēmienu ar Petri tīklu palīdzību nespēju apstrādāt sinhronizācijas saites un biznesa procesa uzvedību kļūdu gadījumos.

Lai arī darba [9] autori min, ka, izstrādājot transformāciju no BPEL uz YAWL, transformācija pretējā virzienā ir pašsaprotama un vienkārša (bezmaz vai vienkārši apgriezta), šī raksta autors vēlas piebilst, ka šis apgalvojums ir patiess tikai tad, ja YAWL darbplūsma pirms tam ir transformēta no BPEL procesa ar minēto rīku, saglabājot BPEL nepieciešamos elementus, vai arī YAWL darbplūsma jau no paša sākuma ir izstrādāta tieši tādam mērķim – to pēc tam transformēt uz BPEL procesu.

Autora piedāvātā transformācija no sākuma vienkāršo YAWL darbplūsmu, bet pēc tam veic šablonu atpazīšanu, kurus pēc tam pārveido uz BPEL aktivitātēm. Līdzīgs uzdevums - šablonu atpazīšana un to pārveidošana uz BPEL konstrukcijām ir aprakstīts darbā [10], taču tas ir balstīts uz BPMN procesa pārveidošanu par Petri tīklu un tālāku tīkla transformāciju uz BPEL, kamēr autora piedāvātā pieeja izmanto no valodas neatkarīgu 'protostruktūru' - procesa plūsmas pierakstu kā orientētu grafu, saglabājot plūsmas semantisko jēgu. Jāatzīmē, ka darbā [10] aprakstītie transformācijas paņēmieni ļauj iegūt BPEL arī no pareizi nestrukturēta BPMN procesa, izmantojot BPEL mehānismu „Notikums-Darbība", taču tad iegūtais BPEL ir samērā nelasāms. Šī raksta autora vairāk sliecas uz nestrukturētas 'protostruktūras' pārveidošanu uz strukturētu 'protostruktūru' (piemēram, izmantojot algoritmus no [11]), jo tad principā ir iespējama procesa transformācija uz jebkādu strukturētu valodu, ne tikai BPEL.

Pāreja no YAWL EWF tīkla uz 'protostruktūru'

Lai atvieglotu pāreju no YAWL darbplūsmas uz BPEL procesu, darbplūsmu aprakstošais grafs tiek vienkāršots, rezultātā izveidojot 'protostruktūru' – orientētu grafu ar viena veida virsotnēm. Protostruktūra tiek izveidota, no YAWL tīkla izvācot virsotnes, kas apzīmē vietas. Nosacījums, kas bija piekārtots izvāktajai vietai, tiek piekārtots jaunajai arkai starp atbilstošajām 'protostruktūras' virsotnēm. Vienīgais izņēmums ir darbplūsmas beigu vieta, kura arī tiek iekļauta 'protostruktūrā'.

Definēsim YAWL darbplūsmu kā (P, T, W), kur:

P – galīga vietu kopa un T – galīga pāreju kopa tā, ka $P \cap T = \emptyset$, bet $W \subseteq (P \times T) \cup (T \times P)$ ir vietas un pārejas savienojošo arku kopa tā, ka neviena pāreja nav savienota ar citu pāreju un neviena vieta nav savienota ar citu vietu. Ar $P_b \subset P$ apzīmēsim visas darbplūsmas beigu vietas – vietas, no kurām darbplūsmā nav definētu izejošo arku: $W \cap (P_b \times T) = \emptyset$. Vienkāršs darbplūsmas piemērs tika dots 1. attēlā.

'Protostruktūru' definēsim kā (A, W), kur:

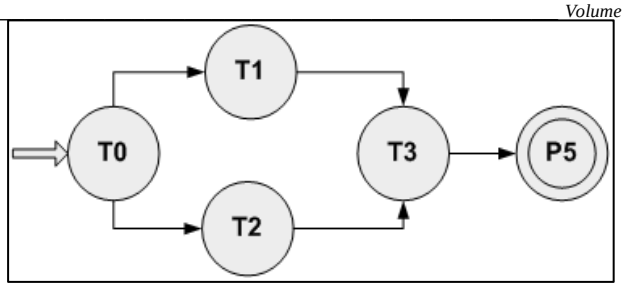
A – galīga darbību kopa tā, ka A satur visus elementus no darbplūsmas pāreju kopas T un visus elementus no darbplūsmas beigu vietu kopas P_b: $A = (T \cup P_b)$.

Ja ar $T_0 \overset{cond}{\Longrightarrow} P_n \Rightarrow T_1$ apzīmē darbplūsmas virzību no pārejas T_0 uz pāreju T_1 caur vietu P_n, kad to atļauj nosacījums $cond$, tad 'protostruktūrā' tiek definēta atbilstoša pāreja no A_0 uz A_1 – $A_0 \overset{cond}{\Longrightarrow} A_1$.

Ja ar $T_m \overset{cond}{\Longrightarrow} P_n$ apzīmē darbplūsmas virzību no pārejas T_m uz vietu P_n, kad to atļauj nosacījums $cond$ un $P_n \in P_b$, tad 'protostruktūrā' tiek definēta atbilstoša pāreja no A_m uz A_n – $A_m \overset{cond}{\Longrightarrow} A_n$. Protostruktūras' piemērs redzams 5.attēlā.

Pāreja no YAWL darbplūsmas uz 'protostruktūru' tiek veikta, analizējot YAWL darbplūsmas pierakstu XML formā.

'Protostruktūra' realizēta ar divu klašu palīdzību, to apraksti doti 1. un 2. tabulā. Klase „Process" apraksta procesu, savukārt klase „Activity" apraksta dotā procesa aktivitātes.



5.att. YAWL darbplūsmai atbilstoša 'protostruktūra'

1. TABULA
KLASES 'PROCESS' ATRIBŪTU UN OPERĀCIJU APRAKSTS

Atribūts/operācija	Apraksts
Name	'Protostruktūras' nosaukums
StartActivity	Sākotnējā 'protostruktūras' virsotne (Activity datu tips)

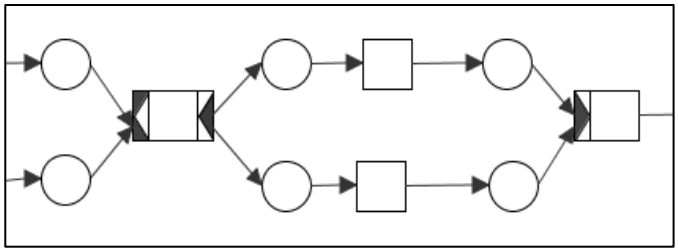
2. TABULA
KLASES 'ACTIVITY' ATRIBŪTU UN OPERĀCIJU APRAKSTS

Atribūts/operācija	Apraksts
Condition	Nosacījums, pie kāda plūsmas virzība iet caur šo virsotni
IncomingType	Ienākošo arku tips: • None – nav ienākošo arku (sākuma virsotnei) • Single – viena ienākošā arka • And – vairākas ienākošās arkas, paralēlo plūsmu sinhronizācija • Xor – vairākas ienākošās arkas, izvēles plūsmu sinhronizācija
Index	Indekss, izmanto ciklu identificēšanā ar Tarjan SCC algoritmu
LowLink	Saite, izmanto ciklu identificēšanā ar Tarjan SCC algoritmu
Name	Virsotnes nosaukums
OutgoingType	Izejošo arku tips: • None – nav izejošo arku (beigu virsotnei) • Single – viena izejošā arka • And – vairākas izejošās arkas, paralēlo plūsmu sadale • Xor – vairākas izejošās arkas, izvēles plūsmu sadale
AddToNextTasks()	Operācija paredzēta, lai no virsotnes sasniedzamo kaimiņu kopai pievienotu jaunu virsotni
ClearNextTasks()	Operācija paredzēta, lai iztīrītu no virsotnes sasniedzamo kaimiņu kopu, tas ir, dzēstu visas izejošās arkas
GetNextTasks()	Operācija paredzēta, lai nolasītu no virsotnes sasniedzamo kaimiņu kopu
NextTasksContains()	Operācija paredzēta pārbaudei, vai no virsotnes sasniedzamo kaimiņu kopa jau nesatur pārbaudāmo virsotni

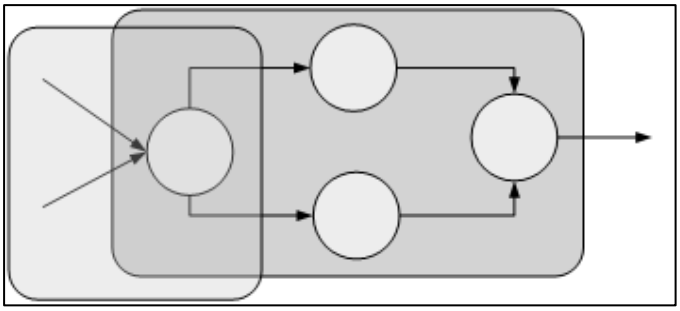
Pāreja no 'protostruktūras' uz BPEL procesu.

BPEL process ir strukturēta, secīgi izpildāmu darbību kopa. Tajā nav pieļaujami patvaļīgi cikli, 'goto' tipa konstrukcijas, procesa konstrukcijas ir vai nu secīgas, vai viena otru iekļaujošas. Tas nozīmē, ka nav pieļaujama situācija, ka BPEL procesa aprakstā tiek atvērta konstrukcija A, piemēram, while, tad konstrukcija B (flow) un tad tiek aizvērta konstrukcija A, bet B tiek aizvērta pēc tam. Īsāk sakot, konstrukciju atvēršana/aizvēršana notiek pēc LIFO principa.

Ir gadījumi, kad 'protostruktūras' virsotnes šo principu neievēro – tas notiek, ja gan ienākošo, gan izejošo arku skaits ir lielāks par 1, tas ir, šī virsotne darbplūsmā reizē gan sinhronizē plūsmu (join), gan to uzreiz arī sadala (split). Šādos gadījumos atbilstošā 'protostruktūras' virsotne tiek sadalīta divās virsotnēs – viena atbild par plūsmas sinhronizāciju, bet otra – par tās sadali. Šāda sadalīšana nepieciešama, lai izveidotajā 'protostruktūrā' būtu iespējams korekti identificēt izpildes konstrukcijas, kā arī to sākuma un beigu virsotnes. 6. attēlā redzams YAWL darbplūsmas fragments, kurā pāreja vienlaicīgi gan sinhronizē plūsmu, gan to sadala.



6.att. Plūsmas sinhronizācija un sadale YAWL darbplūsmā



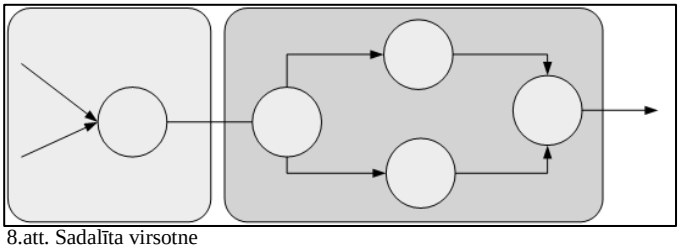
7.att. Konstrukciju pārklāšanās ‘protostruktūrā’

7. attēlā redzama šāda situācija ‘protostruktūrā’. Sadalīšanu veic, apstaigājot visas ‘protostruktūras’ virsotnes un sadalot tās, kurām gan sinhronizējošo, gan sadalošo arku skaits ir lielāks par 1. Sākotnējā virsotne saglabā savas ienākošās (sinhronizējošās) arkas, savukārt izejošās (sadalošās)arkas tiek pievienotas jaunajai virsotnei. Starp abām virsotnēm tiek izveidota jauna arka ar virzienu no sākotnējās uz jauno.

Ja $I(A)$ – virsotnes A ienākošo arku kopa, $O(A)$ – virsotnes izejošo arku kopa, $|I(A)| > 1 \wedge |O(A)| > 1$, tad definējam virsotni A':

- $O(A') = O(A)$ – jaunās virsotnes izejošo arku kopa sakrīt ar vecās virsotnes izejošo arku kopu;
- $O(A) = I(A') = \{A \Rightarrow A'\}$ – vecās virsotnes izejošo arku kopu aizstāj ar vienu arku no vecās virsotnes uz jauno, tā reizē ir arī jaunās virsotnes ienākošo arku kopa.

Papildus virsotne tiek ieviesta arī gadījumos, ja sākuma virsotnes izejošo arku skaits ir lielāks par vienu, lai nodrošinātu BPEL procesa sākšanos ar ‘Receive’ aktivitāti’. 8. attēlā redzama ‘protostruktūra’ ar sadalītu virsotni un skaidri nodalītiem funkcionālajiem blokiem.



8.att. Sadalīta virsotne

Pēc ‘protostruktūras’ papildināšanas to var sākt apstrādāt pa atsevišķiem blokiem. Galvenās ‘protostruktūras’ bloku konstrukcijas, kas tiek apstrādātas, parādītas 3. tabulā.

3. TABULA
APSTRĀDĀJAMO KONSTRUKCIJU APRAKSTS

Konstrukcija	BPEL elements	Apraksts
Vienkārša plūsma	Sequence	Plūsma, kas var saturēt vienkāršas, atomāras darbības un citas plūsmas.
Cikls	While, Repeat-Until	Plūsma, kuras iekšienē definētās plūsmas tiek atkārtotas tik ilgi, kamēr cikla nosacījums ir patiess.
Paralēlā plūsma	Flow	Plūsma, kuras iekšienē ir definētas vismaz divas plūsmas, kuru darbība ir paralēla un nav atkarīga viena no otras. Šīs plūsmas darbība tiek uzskatīta par pabeigtu tad, kad ir izpildītas visas tās iekšienē definētās plūsmas.
Izvēle	If	Plūsma, kuras iekšienē ir definētas vismaz divas plūsmas, kuru darbība ir izslēdzoša, tas ir, šīs plūsmas ietvaros var tikt izpildīta tikai viena plūsma, kuras izpildi nosaka plūsmas nosacījums.

'Protostruktūra' tiek apstrādāta, sākot ar tās sākuma virsotni. Šī aktivitāte tiek pārveidota par BPEL 'Receive' aktivitāti. Uzreiz pēc 'Receive' tiek ievietota arī 'Reply' aktivitāte. Pēc tam rekursīvi tiek apstrādāti šīs virsotnes kaimiņi – dziļumā līdz virsotnei, kas atbilst kā beigu virsotne konstrukcijai, kuru iesāka apstrādes bloka sākotnējā virsotne. Bloka sākuma virsotnei atbilstošo beigu virsotni atrod pēc sekojoša algoritma – ciklā apstrādā katras virsotnes pirmo kaimiņu un pārbauda, vai tā ieejošo arku tips sakrīt ar bloka sākuma virsotnes izejošo arku tipu. Lai izvairītos no kļūdām, kad, teiksim, IF blokā ir ievietots vēl viens IF bloks un lai iekšējā bloka beigu virsotni neuztvertu kā visa bloka beigu virsotni, tiek izmantots konstrukciju skaitītājs – bloka pārbaude beidzas tikai tad, kad tas ir 0. Cikli 'protostruktūrā' tiek identificēti, izmantojot Tarjan SCC algoritmu [12]. Rezultāts ir ciklu saraksts, kurā katrs cikls satur norādes uz savām aktivitātēm. Cikliskās aktivitātes BPEL procesa izveidošanā tiek apstrādāti tāpat kā citas pamatkonstrukcijas – katru sākuma virsotnes kaimiņu rekursīvi dziļumā līdz bloka beigu virsotnei.

Prasības pret YAWL darbplūsmu.

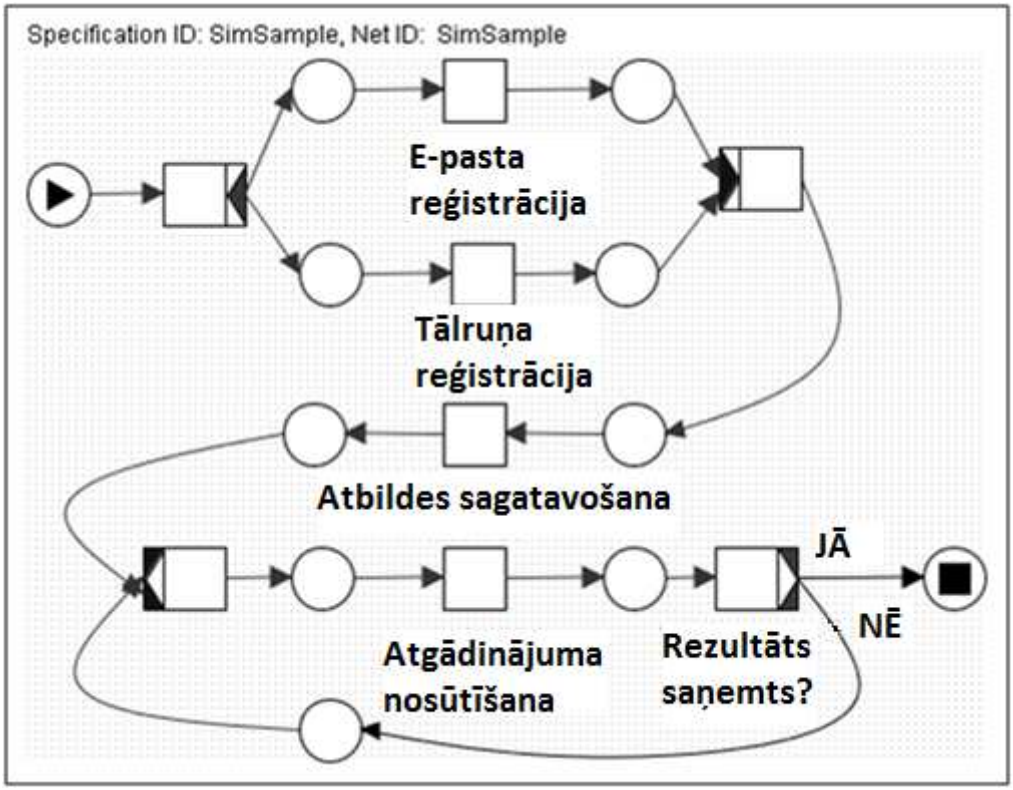
Lai sekmīgi transformētu YAWL darbplūsmu uz BPEL procesu, darbplūsmai jāatbilst zināmām prasībām. Pirmkārt, tā nevar saturēt konstrukcijas, kurām nav analogu starp BPEL konstrukcijām, piemēram, patvaļīgu pāreju no vienas konstrukcijas iekšienes uz citu konstrukciju, apejot sinhronizācijas elementu – tā saucamo 'goto' operatoru. No 20 standarta konstrukcijām, kas minētas darbā [13], BPEL atbalsta 13.

Otrkārt, izmantojot BPEL, saņemtie un izejošie ziņojumi ar procesa instanci tiek saistīti, izmantojot korelācijas kopas. YAWL šāda jēdziena nav, jo katru darbplūsmas instanci (case) manuāli uzsāk tās klienti (lietotāji), līdz ar to arī izveidojot instanci izpildāmajā vidē. Šī vide pārvalda darblūsmu virzību un piedāvā lietotājiem atbilstošas iespējas, balstoties uz darbplūsmas statusu un to specifikāciju [15]. Treškārt, lietotāja uzdevumu atbalsts – BPEL visas aktivitātes, kas saistītas ar procesa apstrādājamo datu iegūšanu no vai nodošanu iesaistītajiem partneriem, uztver kā tīmekļa servisu operācijas, t.i., BPEL nav jēdziena „Lietotāja aktivitāte". Šādu aktivitāšu realizācijai tiek izstrādāti BPEL paplašinājumi, piem., BPEL4People [16] (OASIS izstrādā standartu, kamēr IBM savā risinājumā jau piedāvā atbilstošu implementāciju [18]).

Darbplūsmas definīcijā jābūt korekti aprakstītiem visiem zarošanās nosacījumiem, lai transformācijas rezultātā radītā BPEL procesa WHILE, REPEAT/UNTIL un IF blokos būtu pareizas vērtības.

IV. PIEMĒRS

Simulācijas piemērs attēlo vienkāršu procesu – klientam tiek piedāvāts ievadīt savu kontaktinformāciju (e-pasta adresi un tālruņa numuru), pieprasījums tiek apstrādāts un klientam tiek nosūtīts uzaicinājums saņemt rezultātu. Ja kādā noteiktā laika posmā klients to neizdara, uzaicinājums tiek nosūtīts vēlreiz. Dotajā piemērā ir atrodamas 3 iepriekš minētās projektēšanas problēmas – tālāk tiks noskaidrots, vai simulācija tās identificēs – paralēlā plūsma, resursu trūkums (šaurā vieta) un bezgalīgais cikls (sk.9. attēlu).



9.att. YAWL – simulācijas piemērs

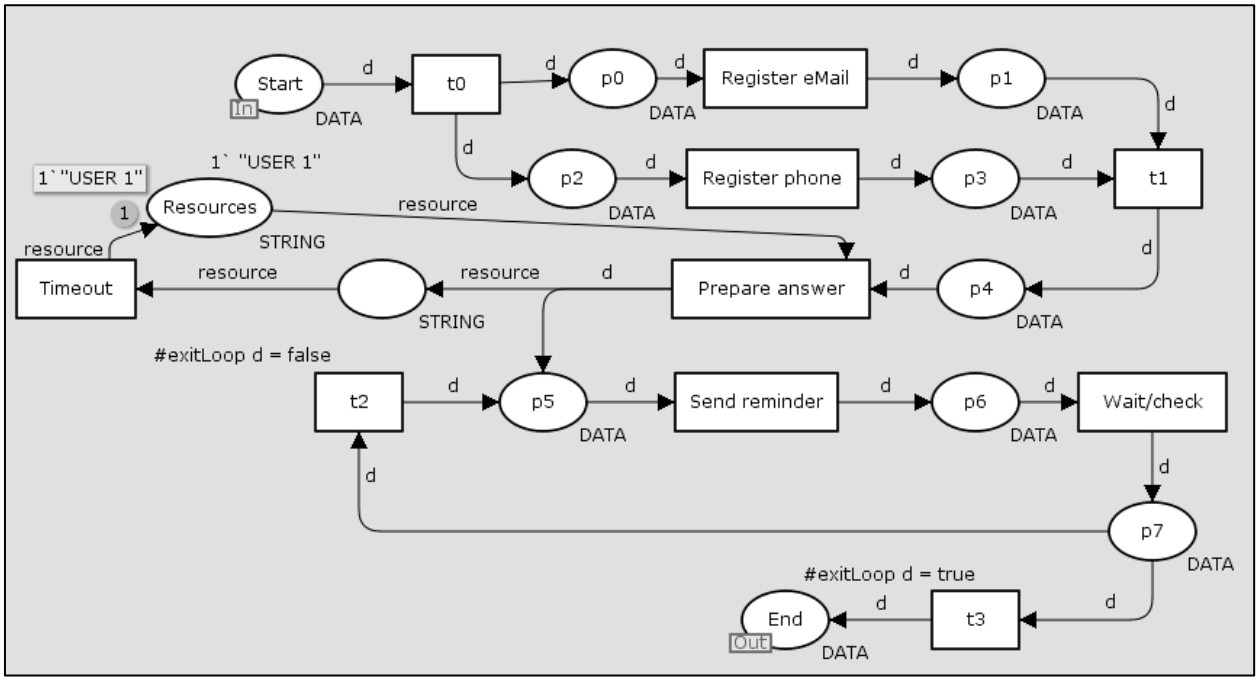
Tiek izveidotas 12 procesa instances – 6 no tām tiek norādīts gan tālruņa numurs, gan e-pasta adrese, bet 6 – tikai e-pasta adrese.

Kā jau iepriekš minēts, ar CPN Tools nav iespējama OR-plūsmu simulācija, tamdēļ 6 procesa instanču nonākšanu strupceļā var konstatēt tikai ar ProM analīzes rīkiem.

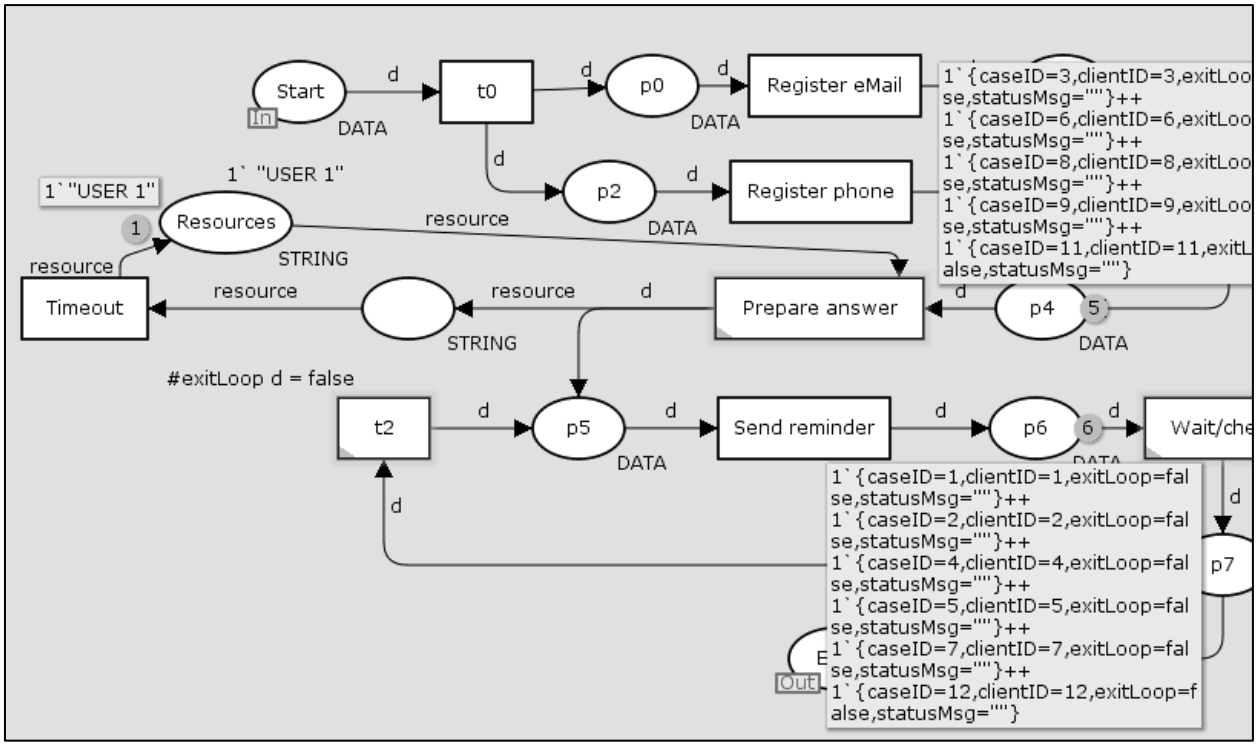
YAWL darbplūsmai atbilstošais CPN tīkls redzams 10. attēlā.

Šaurā vieta procesā ir uzdevumā „Atbildes sagatavošana" – gan YAWL darbplūsmā, gan Petri tīklā tā izpildei ir piesaistīts tikai viens resurss („USER 1"), tāpēc procesu instances gaida, kamēr tas būs pieejams (sk. 11. attēlu – tīkla vieta p4). Pāreja „Timeout" ieviesta mākslīgai aizturei, kas parāda resursa noslodzi – to iespējams darīt, norādot TIMED pie datu tipiem, taču uzskatāmības labad tas netiek darīts.

Pēdējā tīkla problēma ir bezgalīgā cikla bloks – veicot simulāciju, novērojams, ka visas 12 procesa instances nekad nenonāk līdz beigām.



10.att. YAWL darbplūsmai atbilstošais Petri tīkls



11.att. Petri tīkls simulācijas laikā

V. SECINĀJUMI

Autora piedāvātā pieeja biznesa procesu modelēšanā uz vienkārša piemēra bāzes, no sākuma to izstrādājot ar akadēmisku valodu, lai būtu iespējams to analizēt ar statistiskiem līdzekļiem, bet pēc tam transformējot uz biznesa valodu ar plašākām iespējām un rīku klāstu, ir diezgan sekmīga. Galvenais transformācijas ieguvums ir izstrādātā 'protostruktūra' un tās apstaigāšanas un šablonu atpazīšanas algoritms, kas principā ļauj darbplūsmu no YAWL pārveidot uz jebkuru citu, hierarhiski organizētu valodu – gan akadēmisku, gan no biznesa vides nākušu. Apstaigāšanas un atpazīšanas algoritms ir izmantojams arī citās valodās aprakstītiem procesiem, tomēr pirms tam šie procesi jāizmaina, lai tie atbilstu 'nepārklāšanās' kritērijiem.

Aplūkotais simulācijas modelis nav vērtējams viennozīmīgi – vienkāršākās no projektēšanas problēmām, piemēram, šauro vietu vai strupceļu identificēšanu, ar šo pieeju ir iespējams atrisināt, taču sarežģītākās, tādas kā klinču atrašana vai kļūdu apstrādes mehānisma pārbaude, atrisināt nav iespējams. Tas gan vairāk ir saistīts ar to, ka izvēlētais simulācijas modelis ir balstīts uz krāsainajiem Petri tīkliem, kuros nav iespējams definēt kļūdu apstrādes mehānismus. Iespējams, ka balstot simulāciju un sākotnējo procesa izstrādi uz kādu citu valodu, varētu tikt atrisinātas arī citas iepriekš minētās projektēšanas problēmas.

Atsaucoties uz autora darbu [20], piedāvātā transformācija ir sekmīgi atpazinusi YAWL darbplūsmā izmantotos šablonus un atveidojusi procesa struktūru, taču nav izveidotas vairākas svarīgas procesu aprakstošās sadaļas. Daļu no tām nav iespējams izveidot, izmantojot autora piedāvāto pieeju – tas attiecas uz sadaļu „faultHandlers", jo izmantotais simulācijas modelis, kas balstīts uz Petri tīkliem, neatbalsta kļūdu apstrādes mehānismu simulāciju. Citas no tām vienkārši nav apstrādātas – „variables" un „partnerRoles", savukārt citas, piemēram, „correlationSets", nevar izveidot vispār. Šī piemēra ietvaros netika pārbaudīts, kā transformācija apstrādātu sarežģītākus darbplūsmu šablonus, piemēram, „foreach".

Apkopojot projektēšanas rezultātus, autors secina, ka rakstā piedāvātā pieeja jāattīsta tālāk – jāspēj automātiski pārveidot sākotnējā 'protostruktūra', lai atbrīvotos no pārklāšanās gadījumiem situācijās, kad izmantoti sarežģītāki darbplūsmu šabloni, piemēram, goto tipa konsrukcijas. Tāpat arī jāizpēta citi simulācijas paņēmieni, lai atrisinātu tās projektēšanas problēmas, kuras nespēja apstrādāt aplūkotais simulācijas modelis, savukārt transformācija uz BPEL jāpapildina ar iespēju izveidot arī sadaļas „variables" un „partnerRoles", kā arī jādefinē prasības pret sākotnējo YAWL modeli, lai būtu iespējams automātiski izveidot arī sadaļas „faultHandlers" un „correlationSets".

LITERATŪRAS SARAKSTS

[1] M. Weske "Business Process Management", Springer 2007, p. 169.
[2] W. M. P. van der Aalst, A. H. M. ter Hofstede. „YAWL: Yet Another Workflow Language" Inf. Syst., 30(4):245–275, 2005.
[3] C. Ouyang, A.H.M. ter Hofstede, M. La Rosa, M. Rosemann, K. Shortland and D. Court, Camera, Set, Action: Automating Film Production via Business Process Management. In Proceedings of the International Conference "Creating Value: Between Commerce and Commons", Brisbane, Australia, 2008.
[4] YAWL4Health, http://www.yawlfoundation.org/casestudies/health.
[5] N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, and N. Mulyar „Workflow Control-Flow Patterns: A Revised View", BPM Center Report BPM-06-22 , BPMcenter.org, 2006
[6] A. Rozinat, M.T. Wynn, W.M.P. van der Aalst, A.H.M. ter Hofstede, and C. J. Fidge. „Workflow Simulation for Operational Decision Support Using Design, Historic and State Information" Proceedings of the 6th International Conference on Business Process Management (BPM 2008), 2008, Milan, Italy, LNCS 5240, 196 – 211, 2008, Springer-Verlag.
[7] W.M.P. van der Aalst, B.F. van Dongen, C.W. Gunther, R.S. Mans, A.K. Alves de Medeiros, A. Rozinat, V. Rubin, M. Song, H.M.W. Verbeek, and A.J.M.M. Weijters. ProM 4.0: Comprehensive Support for Real Process Analysis. In J. Kleijn and A. Yakovlev, editors, Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2007), volume 4546 of Lecture Notes in Computer Science, pages 484-494. Springer-Verlag, Berlin, 2007.
[8] K. Jensen, L.M. Kristensen, and L. Wells. Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. International Journal on Software Tools for Technology Transfer, 9(3-4):213-254, 2007.
[9] Antonio Brogi, Razvan Popescu „BPEL2YAWL: Translating BPEL processes into YAWL workflows", University of Piza, 2006
[10] Chun Ouyang , Marlon Dumas , Wil M. P. Van Der Aalst , Arthur H. M. Ter Hofstede , Jan Mendling, From business process models to process-oriented software systems, ACM Transactions on Software Engineering and Methodology (TOSEM), v.19 n.1, p.1-37, August 2009
[11] J. Koehler and R. Hauser. Untangling unstructured cyclic flows - A solution based on continuations. In R. Meersman, Z. Tari, W.M.P. van der Aalst, C. Bussler, and A. Gal et al., editors, On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2004, volume 3290 of Lecture Notes in Computer Science, pages 121–138, 2004.
[12] Strong Connectivity, http://www.ics.uci.edu/~eppstein/161/960220.html#sca
[13] P. Wohed, W. van der Aalst, M. Dumas, A. H. M. ter Hofstede, „Pattern Based Analysis of BPEL4WS", FIT Technical Report, FIT-TR-2002-04
[14] M. Havey „Essential Business Process Modeling" O'Reilly 2005, p. 141
[15] W. M. P. van der Aalst, L. Aldred, M. Dumas, T. A. H. M. Hofstede. „Design and Implementation of the YAWL System", „Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAiSE'04)", Riga, Latvia, volume 3084 of LNCS, 142–159, 2004
[16] Holmes T., Vasko M., Dustdar S.: VieBOP: Extending BPEL Engines with BPEL4People. 16th Euromicro International Conference on Parallel, Distributed and network-based Processing 2008, 547-555. February 2008.
[17] BPEL 2.0 specification, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
[18] WS-BPEL Extension for People, http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/
[19] Web Services Flow Language, http://www.doc.ic.ac.uk/~hf1/phd/papers/toread/WSFL.pdf
[20] Stipravietis P., Ziema M. The Design of Electronic Service Process Using YAWL. Proceedings of the Ninth International Baltic Conference Baltic DB&IS 2010, Latvia, Riga, 2010. - pp 75-90.

Peteris Stipravietis earned his master's degree in computer sciences from University of Latvia in 2006. Currently he is a doctoral student at Riga Technical University.
He is working for SIA ZZ Dats, Riga, Latvia as a project manager and system analyst, beginning his career as a programmer.
His main research interest is designing of electronic services, simulation and transformation of underlying business processes and workflows. Last publication in this field was on July, 2010 – "The Design of Electronic Service Process Using YAWL", the 9th Int. Baltic Conference Baltic DB&IS.
Contact information: peteris.stipravietis@rtu.lv, peteris.stipravietis@zzdats.lv

Peteris Stipravietis. The usage of business process simulations and transformations in designing electronic services
The article discusses the solution of common business process design-time problems using YAWL – academic workflow language based on enhanced workflow nets. It currently supports all workflow patterns. The approach proposed by the authors is based on the creation of business process in the YAWL environment in order to simulate and validate the process model which could resolve some of the design-time problems as well as provide hints to correct initial process. Simulation is based on transformation of the YAWL workflow to colored Petri net – the net is then simulated to identify problems such as possible infinite loops, bottlenecks, process waits and others. The simulation methods to identify these problems are discussed. There are some design-time problems which cannot be identified or resolved using simulation approach discussed, for example, cancellation mechanisms and OR-flows, because Petri nets (upon which the simulation is based) lack these concepts. The article also describes technique to acquire the primitive description of process from the YAWL workflow, preserving control and data flow. The primitive description provides next step to authors approach – the transition from academic language to business language, which may lack the semantics, validation and simulation means as opposed to academic languages, but have superior execution environments and developer support. The primitive description is represented as oriented graph and is used to transform the YAWL workflow to a business process described using some business language, in this case BPEL, although the primitive structure may be used as the source of transformation to any hierarchic business process language. The article also contains an example – simple YAWL workflow containing three of possible problems mentioned before. The workflow is then transformed and simulated to determine if simulation approach proposed can identify the problems of sample workflow.

Петерис Стиправиетис. Использование трансформации и симуляции бизнес-процессов в проектирование электронных услуг
В статье рассматривается решение общих проблем во времени разработки бизнес-процесса, используя YAWL – академический язык рабочего потока, основанный на расширенных сетях рабочего потока, который в настоящее время поддерживает все образцы рабочих потоков. Подход, предложенный авторами, основан на создании бизнес-процесса используя YAWL, чтобы симулировать и проверить модель процесса. Эта модель может решить некоторые из проблем во времени разработки. Симуляция основана на преобразовании рабочего потока YAWL к цветной сети Petri – полученная сеть моделируется, чтобы идентифицировать проблемы, такие как возможные бесконечные петли, узкие места и другие. Также в статье обсуждены методы симуляции, чтобы идентифицировать эти проблемы. К сожелению, существует некоторые проблемы, которые не могут быть идентифицированы исползуя обсужденный подход, например, механизмы отмены и ИЛИ-потоки, потому что сети Petri (на которых базируется симуляция) не поддерживают эти понятия. Статья также описывает технику, чтобы приобрести примитивное описание процесса YAWL, сохраняя поток контроля и данных. Примитивное описание обеспечивает следующий шаг – переход от академического языка к бизнес-языку, которому может не хватать средств семантики, ратификации и симуляции по сравнению с академическим языкам, но который имеет более хорошую окружающую среду выполнения и поддержку разработчика. Примитивное описание представлено как ориентированный граф и используется, чтобы преобразовать рабочий поток к бизнес-процессу, описанному используя некоторый бизнес-язык, в этом случае BPEL. Примитивная структура может использоваться в качестве источника преобразования на любой иерархический язык бизнес-процесса. Статья также содержит пример – простой рабочий поток YAWL, содержащий три из возможных проблем, упомянутых прежде. После разработки потока преисходит преобразование и симуляция, чтобы определить, может ли предложенный подход симуляции идентифицировать проблемы данного рабочего потока.