KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Giedrius Naujokaitis

VERSLO TRANSAKCIJŲ ATVAIZDAVIMAS DUOMENŲ BAZĖS TRANSAKCIJOMIS

Magistro darbas

Vadovas
doc. B. Paradauskas

KAUNAS, 2007

SUMMARY

REPRESENTING BUSINESS TRANSACTIONS AS DATABASE TRANSACTIONS

Pragmatically motivated communicative action loop (CAL) is proposed for modeling business transactions and designing database transactions. Using parameters of the communicative action loops, activity model can be adapted to particular objective area. Selected parameters are utilizable in formation of database transactions that are oriented in modification of states of objects of activity. Performed database transactions are registered to query register in order to make a database rollback when business transactions are canceled. Database transactions are synchronized between two replicated databases.

SANTRAUKA

Šiame darbe apibrėžiama pragmatiškai motyvuota komunikacinė veiksmų kilpa, kurią pasiūlyta naudoti verslo transakcijoms modeliuoti ir duomenų bazių transakcijoms projektuoti. Komunikacinės veiksmų kilpų parametrais veiklos modelis adaptuojamas konkrečiai dalykinei sričiai. Parinkti parametrai panaudojami sudarant duomenų bazės transakcijas, orientuotas į veiklos objektų būsenų modifikavimą. Įvykdytos duomenų bazės transakcijos registruojamos ir, nutraukus verslo transakcijas, panaudojamos duomenų bazės atstatymui. Duomenų bazės transakcijų vykdymas sinchronizuojamas tarp dviejų replikuojančių duomenų bazių.

TURINYS

1. ĮVADAS 5
2. VERSLO TRANSAKCIJŲ MODELIAVIMO IR VALDYMO ANALIZĖ 7
2.1. Verslo transakcijų modeliavimas 7
2.1.1. Verslo transakcijų šablonai 8
2.1.2. Verslo transakcijų nuoseklumas 10
2.1.3. Verslo transakcijų modeliavimas komunikacinėmis veiksmų kilpomis 10
2.1.4. Verslo transakcijų būsenos 12
2.2. Verslo transakcijų valdymas 14
2.2.1. Verslo transakcijų klaidų valdymas 14
2.2.2. Duomenų bazės atstatymas po klaidų 15
2.2.3. Verslo transakcijos paskirstytose duomenų bazėse 17
2.3. Verslo transakcijų modeliavimo metodų, standartų ir produktų apžvalga 19
2.3.1. Elektroninių ryšių valdymas ebXML 19
2.3.2. Winograd-Flores komunikacinių kilpų modelis 20
2.3.3. BTP protokolas 22
2.3.4. Elektroninio verslo modelių sudarymo kalba XLANG 22
2.4. Verslo transakcijų modeliavimo ir valdymo išvados 23
3. DUOMENŲ BAZIŲ TRANSAKCIJŲ MODELIAVIMAS NAUDOJANTIS KOMUNIKACINĖMIS VEIKSMŲ KILPOMIS 24
3.1. Komunikacinių veiksmų kilpų sudarymas 24
3.1.1. Statinės priklausomybės 24
3.1.2. Dinaminės priklausomybės 25
3.1.3. Pragmatinės priklausomybės 28
3.2. Verslo transakcijų užduotų komunikacinėmis veiksmų kilpomis parametrų parinkimas 29
3.3. DB struktūrų atitikimo analizė užduotam paslaugos tipui 33
3.4. Duomenų bazės transakcijų sudarymas 34
3.5. Transakcijų realizavimas replikuojančiose DB 36
3.6. Įvykdytų duomenų bazės transakcijų atstatymas 38
4. VERSLO TRANSAKCIJŲ VYKDYMO ELEKTRONINIŲ PASLAUGŲ SRITYJE EKSPERIMENTINIS TYRIMAS 43
4.1. Dalykinės srities apibrėžimas 43
4.2. Eksperimentinis verslo transakcijų vykdymas 43
4.3. Eksperimentinės sistemos architektūra 60

4.4. Sistemos išplėtimo rekomendacijos 62
5. IŠVADOS 63
6. LITERATŪRA 65
7. TERMINŲ IR SANTRUMPŲ ŽODYNAS 67
8. PRIEDAI 68
Mokslinis straipsnis, paruoštas konferencijai Informacinės Technologijos 2007 68

1. ĮVADAS

Informacinės technologijos šiuolaikiniame versle vaidina vis didesnį vaidmenį suteikdamos galimybę verslo partneriams bendradarbiauti elektroniniu būdu. Elektroniniame versle turi būti užtikrintas suderintas ir patikimas informacinių (verslas – verslui (B2B)) sistemų veikimas. Verslas – verslui sistemų pagalba verslo partneriai elektroniniu būdu gali vykdyti verslo procesus, kuriuos būtina tiksliai specifikuoti ir formaliai apibrėžti. Verslo procesai neatsiejami nuo apsikeitimo duomenimis, todėl informacinėse sistemose yra labai svarbus verslo transakcijų palaikymas. Verslo duomenys yra saugomi duomenų bazėse, todėl atsiranda poreikis verslo transakciją susieti su duomenų bazės transakcija.

Darbo tikslas yra sudaryti verslo transakcijų modelį ir verslo transakcijų modelio pagrindu projektuoti duomenų bazės transakcijas, kuriomis būtų palaikomas DB pilnumas ir nuoseklumas. Darbe apžvelgiami įvairūs verslo transakcijų specifikavimo modeliai. Plačiau nagrinėjamas išplėstinių komunikacinių kilpų metodas, kurio pagrindas yra komunikacinės binarinės kilpos, tarpusavyje susietos pragmatinėmis bei semantinėmis priklausomybėmis. Komunikacinės veiksmų kilpos modeliuoja ne tik aktorių atliekamus veiksmus, bet ir verslo transakcijų būsenų kaitą. Verslo transakcijos paprastai būna sudėtinės ir jų dinamiškumui valdyti naudojamos būsenų gyvavimo ciklo (*exist*) priklausomybės.

Norint kompiuterizuoti verslo transakcijas, būtina išanalizuoti ir formaliai aprašyti su transakcijomis susijusių aktorių, veiksmų, informacijos srautų bei būsenų informaciją. Formalizuojant transakcijas jų elementai yra skaidomi į informacijos komponentines grupes. Tokiu būdu surandami verslo transakcijas apibrėžiantys parametrų rinkiniai, kurie detaliai aprašo atitinkamą verslo transakciją. Parinkti parametrų rinkiniai jau gali būti pateikiami kompiuterizuotai sistemai. Vykstant verslo transakcijoms verslo partneriai keičiasi informacija, kuri aprašoma parametrų rinkiniais. Turėdami aiškiai apibrėžtą informaciją galime ją saugoti duomenų bazėje, nes parametrai nusako duomenų struktūrą reikalingą tai informacijai išsaugoti.

Parametrizuotos komunikacinės veiksmų kilpos yra pagrindas duomenų bazės transakcijų sudarymui. Naudojantis komunikacinių kilpų parametrais sudaromos DB užklausos, kurios vykdo verslo transakcijų logiką, o veiksmų nuoseklumą galima kontroliuoti duomenų bazėse saugant verslo objektų būsenas.

Versle vyksta įvairiausių procesų. Vieni jų įvykdomi iki galo, o kiti nutraukiami dėl įvairių priežasčių. Iki verslo proceso nutraukimo momento jau gali būti įvykdyta keletas verslo procesą sudarančių transakcijų. Tokiu atveju reikia atstatyti duomenų bazėje atliktas užklausas. Visoms iki proceso nutraukimo atliktoms DB transakcijoms sudaromos kompensacinės DB transakcijos, kurios

atstato verslo proceso metu atliktas DB modifikacijas. Tam, kad būtų galima atlikti DB atstatymą, visos verslo proceso metu atliktos DB transakcijos turi būti registruojamos.

Dažnai keičiantis ir vis tobulėjant verslui, verslo informacinės sistemos privalo greitai ir lanksčiai prisitaikyti prie atsirandančių pokyčių Keičiantis verslo aplinkai gali keistis ir įmonių teikiamų paslaugų, parduodamų prekių tipai, tuo pačiu keičiantis ir duomenų struktūroms, reikalingoms atsiradusiems pokyčiams palaikyti. Darbe aptariamas duomenų struktūrų atitikimo naujai atsiradusiam paslaugų tipui nustatymo būdas, kuomet atliekamas pokyčių formalizavimas leidžiantis atlikti duomenų struktūrų palyginimą.

Dauguma įmonių turi dideles individualias duomenų bazes, kurios naudojamos įmonės reikmėms. Dalis duomenų bazės gali būti naudojama įmonės vidinei veiklai palaikyti, o kita dalis – verslo transakcijų palaikymui. Jeigu verslo reikalais bendradarbiauja dvi tokios įmonės, turinčios skirtingas duomenų struktūras, tuomet atsiranda DB replikavimo poreikis. Naudojant replikuojančias duomenų bazes verslo partneriai gali realizuoti savo servisus skirtingose aplinkose. Tokiu būdu DB transakcijos yra vykdomos replikuojančiose duomenų bazėse ir atliekama jų sinchronizacija, kurios metu išlaikomas DB pilnumas bei nuoseklumas.

Darbe atliekamas eksperimentinis tyrimas pasirenkant konkrečią elektroninių paslaugų sritį - internetinių svetainių kūrimą. Internetinės svetainės kūrimo proceso metu klientas įmonei pateikia užsakymą, kuriame surašo savo pageidavimus kuriamai sistemai. Įmonei priėmus užsakymą sudaroma sutartis. Pasirašius sutartį pradedamas vykdyti projektas, kurio metu abi šalys gali bendrauti tarpusavyje viena kitai siųsdamos pranešimus. Kai projektas užbaigiamas ir patvirtinamas, pateikiama sąskaita ir vykdomas apmokėjimas. Eksperimente didžiausias dėmesys kreipiamas į išplėstinėmis komunikacinėmis veiksmų kilpomis specifikuotų verslo transakcijų realizaciją replikuojančiose duomenų bazėse, kuomet atliekamas transakcijų formalizavimas bei sudarytų DB užklausų sinchronizuotas vykdymas.

Magistro darbo tema paruoštas straipsnis konferencijai „Informacinės Technologijos 2007“.

2. VERSLO TRANSAKCIJŲ MODELIAVIMO IR VALDYMO ANALIZĖ

2.1. Verslo transakcijų modeliavimas

Verslo partneriai vis dažniau bendradarbiauja elektroniniu būdu. Gali būti tiek dvišalis bendradarbiavimas, tiek daugiašalis. Šiame darbe nagrinėjamas tik dvišalis bendradarbiavimas.

Dvišalio bendradarbiavimo atveju verslo partneriai vykdo verslo procesus, kurie leidžia abiem šalims pasiekti savo tikslų. Verslo procesai susideda iš aibės tam tikra tvarka vienas po kito vykdomų veiklų - verslo transakcijų, kurios gali naudotis keletu duomenų bazių. Tam tikros srities versle vykdomos transakcijos yra tarpusavyje susijusios. Kiekviena transakcija vienaip ar kitaip keičia verslo santykių būseną tarp bendradarbiaujančių partnerių. Verslo transakcijų metu bendradarbiaujančios šalys keičiasi verslo informacija (pranešimais), kuri vis papildo vykstantį verslo procesą ir veda jį į užbaigimo būseną atitinkančią verslo partnerių siekiamus tikslus. Verslo transakcijos formuoja verslo proceso specifikacijos schemą (*BPSS – Business Process Specification Schema*), kurioje apibrėžiamos visos verslo proceso vykdymo charakteristikos, galimi scenarijai, apibrėžiantys įvykių sekas ir reikalavimus paslaugoms [1].

Nėra bendro formalaus verslo transakcijos apibrėžimo, tačiau verslo transakcija suprantama kaip verslo informacijos ir verslo signalų apsikeitimas tarp dviejų verslo partnerių [2]. Kitaip sakant verslo transakcija yra darbo (veiksmų) tarp dviejų verslo partnerių vienetas [3], susidedantis iš vieno verslo partnerio inicijuojamo veiksmo ir kito verslo partnerio atsako. Jeigu vis dar laukiame verslo partnerio atsako, tai verslo transakcija dar neužbaigta [4]. Šiais veiksmais partneriai siekia realizuoti savus tikslus. Visą verslo procesą galima traktuoti kaip sudėtinę verslo transakciją, kuri sudaryta iš kitų verslo transakcijų - subtransakcijų. Taigi verslo transakcija yra dvišalis bendradarbiavimas, kuris daugiau negali būti dekomponuotas [4]. Tačiau iškyla problema kuomet reikia nustatyti ar verslo transakcija jau yra dekomponuota ir jai įvykdyti nereikia kitų verslo transakcijų. Taigi verslo transakcijų modeliavimas yra sudėtingas procesas reikalaujantis modeliuojamos dalykinės srities žinių ir turi būti atliekamas atitinkamą dalykinę sritį išmanančio specialisto, kuris žinotų visus verslo proceso vykdymo aspektus. Pavyzdžiui, jeigu verslo procesas yra užsakymas, tuomet turi būti detaliai specifikuota visa informacija apie patį užsakymą (apmokėjimas, pristatymas ir kita). Turi būti atsakyta į klausimus: *kas*, *ką*, *kada*, *kur* ir *kodėl*. *Ką* elektroninio verslo terminais paprastai reiškia produktą, t.y. prekes arba paslaugas [5].

2.1.1. **Verslo transakcijų šablonai**

Modeliuojant verslo transakcijas kiekviena jų yra susiejama su vienu iš šešių konkrečių verslo transakcijų šablonų [3]:

- Komercinė transakcija – paprastai šis šablonas naudojamas kaip formalus verslo partnerių įsipareigojimas.
- Pranešimas – naudojamas formaliems verslo pranešimams tokiems kaip verslo transakcijos vykdymo klaidos pranešimas, kuris gali būti susijęs su komercine transakcija, kai, pavyzdžiui, nevykdomi šalių įsipareigojimai.
- Informacijos siuntimas – tai neformalus informacijos apsikeitimas tarp bendradarbiaujančių šalių.
- Užklausa / Atsakymas (atsakymas į užklausą) – naudojama šalies, kuri prašo informacijos iš kitos šalies, kuri jau turi tą informaciją.
- Prašymas / Patvirtinimas (prašymo, reikalavimo (*request*) patvirtinimas) – naudojama kai viena šalis prašo jos būsenos, susijusios su ankstesniais įsipareigojimais arba atsakovo verslo taisyklėmis, patvirtinimo iš kitos šalies.
- Prašymas / Atsakymas (atsakymas į prašymą, reikalavimą) – naudojama kai viena šalis iš kitos prašo informacijos, kurią kita šalis jau turi ir kai verslo informacijos prašymas reikalauja tarpusavyje susijusių rezultatų.

Verslo transakcijų šablonai turi skirtingą semantiką ir tam tikro šablono transakcijoms keliami skirtingi reikalavimai. Verslo transakcijų šablonai tarpusavyje palyginami *1 lentelėje*:

1 lentelė. Verslo transakcijų šablonai

Verslo proceso šablonas	Atsakymas	Formalus / neformalus	Pavyzdys
Komercinė transakcija	Taip	Formalus	Pirkėjas pageidauja pardavėjo per tam tikrą laiką pristatyti produktą į iš anksto sutartą vietą. Pripažindamas savo įsipareigojimus pardavėjas sutinka ir įvykdo prekių pristatymą taip užbaigdamas verslo transakciją.
Pranešimas	Ne	Formalus	Šalis, kuri yra nepatenkinta verslo procesu, nutraukia verslo transakciją ir siunčia pranešimą atšaukdama visą

			verslo procesą, kuriuo buvo nepatenkinta.
Informacijos siuntimas	Ne	Pagal susitarimą	Pardavėjas informuoja savo pirkėjus apie naujo produkto linijos išleidimą, kuri tapo produktų katalogo dalimi. Kai pirkėjai parsisiunčia ir išsisaugo prekių katalogą, jie gali patvirtinti gavimą.
Prašymas / Atsakymas	Taip	Pagal susitarimą	Pirkėjas teiraujasi pardavėjo apie tam tikro produkto kainą ir galimybę jį įsigyti. Užklausa nesibaigia tuo, kad produktas rezervuojamas pardavimui ateityje. Pardavėjas peržiūri savo tiekimo grandinės valdymo ir inventoriaus sistemas tam, kad pateiktų pakankamai išsamų atsakymą. Pardavėjas turi apskaičiuoti kainą priklausančią nuo tiekėjų ir t.t. Dažniausiai atsakymas nesibaigia Taip/Ne atsakymu
Prašymas / Patvirtinimas	Taip	Pagal susitarimą	Pirkėjas teiraujasi pardavėjo, ar jis vis dar turi teises pardavinėti tam tikrą produktą. Pirkėjas tikisi patvirtinimo. Atsakymas neprilyginamas įsipareigojimams, nes tarp šalių gali dar ir nebūti pasirašyta sutartis. Pardavėjas patvirtina, kad jis vis dar turi teises pardavinėti produktą. Paprastai tai būna Taip/Ne atsakymas.
Užklausa / Atsakymas	Taip	Pagal susitarimą	Pirkėjas teiraujasi pardavėjo apie tam tikro produkto kainą ir galimybę jį įsigyti. Užklausa nesibaigia tuo, kad produktas rezervuojamas pardavimui ateityje. Pardavėjas turi kompiuterinį produktų katalogą ir gali pateikti pirkėjui atsakymą be jokių apribojimų.

2.1.2. **Verslo transakcijų nuoseklumas**

Kaip buvo minėta verslo transakcijos vyksta tam tikra tvarka. Taigi, turi būti iš anksto apibrėžta verslo transakcijų vykdymo tvarka ir perėjimai nuo vienos transakcijos prie kitos. Patogu vykdymo tvarką vizualiai specifikuoti naudojant veiklos diagramas, kurios turi pradžios būseną, įvykdymo (pabaigos) būseną, išsišakojimus, sujungimus, perėjimus tarp veiklų ir perėjimų sąlygas [6]. Be veiklos diagramų verslo proceso eigą vizualiai pateikti galima naudojant ir kitas notacijas tokias kaip verslo procesų modeliavimo notacija (BPMN - *Business Process Modeling Notation*). Verslo proceso transakcijų vykdymo eigą galima specifikuoti ir naudojantis klasių diagramomis kartu su sekų diagramomis arba panaudojimo atvejų (*Use Case*) diagrama su detaliomis panaudojimo atvejų specifikacijomis. Modeliavimui naudojant panaudojimo atvejų diagramą galima išskirti tokius pagrindinius modeliavimo žingsnius [4]:

- Verslo procesų identifikavimas – apibrėžiami panaudojimo atvejai;
- Panaudojimo atvejų detalizavimas – apibrėžiamos verslo transakcijos, jų informacijos srautai, vykdymo sąlygos ir vidinės būsenos;
- Verslo informacijos ir paslaugų (servisų) apibrėžimas, kuomet realizuojamos verslo transakcijos aibe kompiuterizuotos informacinės sistemos palaikomų funkcijų (mūsų atveju funkcijos reiškia duomenų bazių transakcijas).

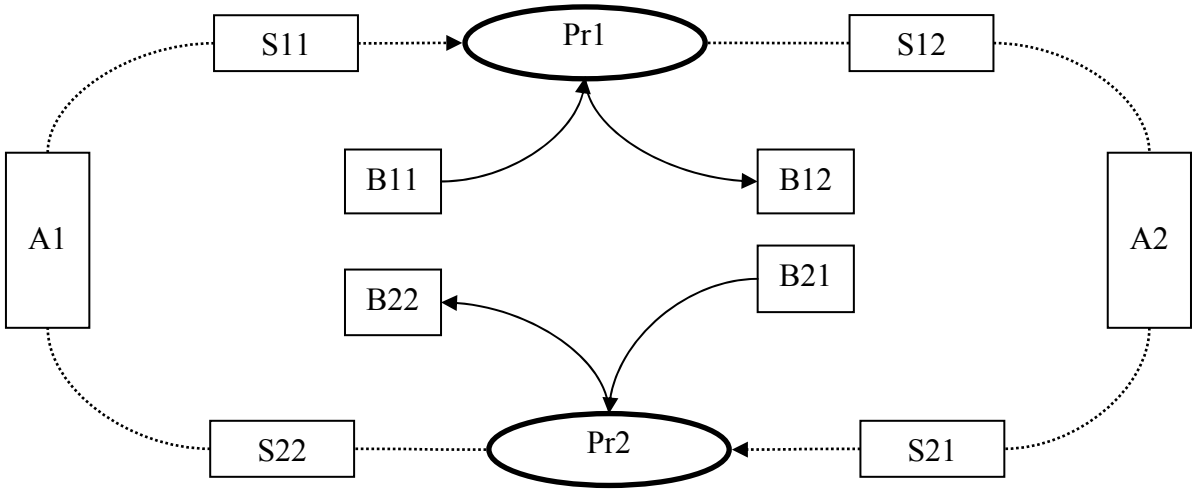
Tačiau šių trijų žingsnių neužtenka pilnai verslo proceso specifikacijai. Reikia apibrėžti ankščiau minėtą verslo transakcijų vykdymo tvarką. Apibrėžiant vykdymo tvarką įvedama verslo proceso būsena, kuri keičiasi vykstant verslo transakcijoms. Taigi verslo transakcijų tvarka specifikuojama verslo proceso būsenomis ir tų būsenų kaita. Verslo partneriai ir procesas prieš verslo transakciją yra vienoje apibrėžtoje būsenoje, o po verslo transakcijos - kitoje apibrėžtoje būsenoje. Jeigu verslo procesas perėjo į tam tikrą būseną, tai dar nereiškia, kad iškart turi prasidėti veiksmas, kurio *prieš būsena* yra tenkinama. Verslo būsenos pasikeitimas reiškia, kad veiksmas jau yra „aktyvuotas“ (jau galimas) ir gali būti vykdomas kai verslo partneris bus pasiruošęs atkreipiant dėmesį į laiko apribojimus ir jų neviršijant.

2.1.3. **Verslo transakcijų modeliavimas komunikacinėmis veiksmų kilpomis**

Patogu verslo transakcijas modeliuoti naudojant komunikacines veiksmų kilpas (Enterprise Modeling (EM) metodas). Šiuo būdu verslo transakcijos apibrėžiamos kaip komunikacinė veiksmų kilpa (*Communicative Action Loop* arba *CAL*) [4], [17]. Bendru atveju verslo transakcija gali būti

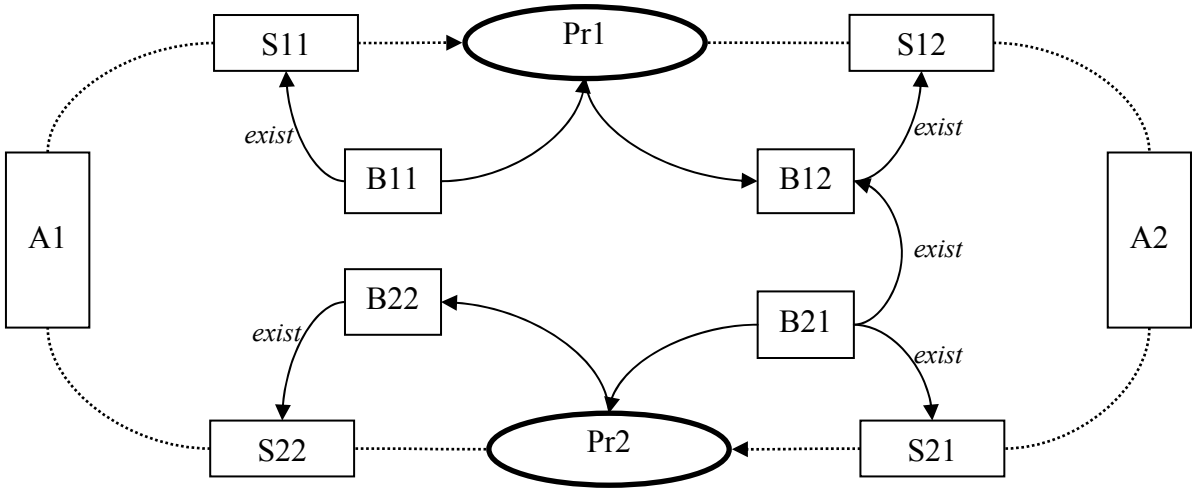
sulyginta su panaudojimo atveju, kaip eilė atliekamų veiksmų, duodančių tam tikrą rezultatą verslo partneriui (aktoriui), kuris tuos veiksmus inicijavo.

1 paveiksle pateikta pavyzdinė komunikacinė (binarinė) veiksmų kilpa sudaryta iš dviejų aktorių – kliento (A1) ir atlikėjo (A2), pranešimų srautų S11÷S22, ir dviejų veiksmų – kliento veiksmo (Pr1) ir atlikėjo veiksmo (Pr2). Dinaminiai situacijos pasikeitimai prieš ir po veiksmų pateikiami kaip perėjimai nuo būsenų B11, B21 į būsenas B12, B22 ir aibė pragmatinių bei semantinių priklausomybių tarp jų.



1 pav. Bazinė komunikacinė veiksmų kilpa

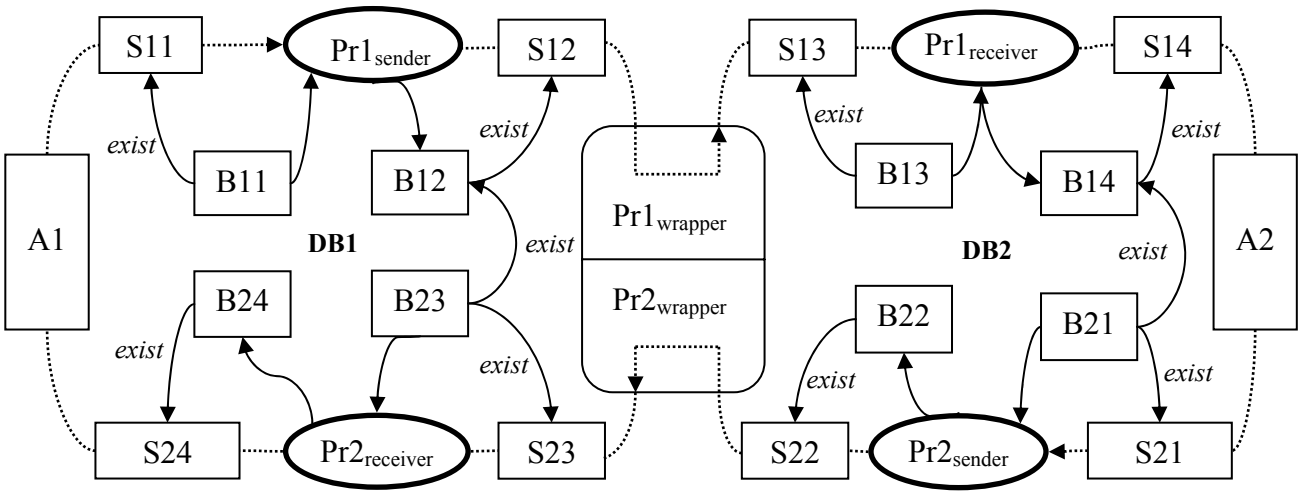
2 paveiksle pateikta išplėsta komunikacinė veiksmų kilpa.



2 pav. Išplėsta komunikacinė veiksmų kilpa

Veiksmai Pr1 ir Pr2 atlieka būsenų pasikeitimą. Vykstant Pr1 veiksmui būsena pasikeičia iš B11 į B12, o vykstant Pr2 veiksmui būsena pasikeičia iš B21 į B22. Galinės būsenos sulyginamos su aktorių tikslais.

3 paveiksle pavaizduota dekomponuota išplėsta komunikacinė veiksmų kilpa. Verslo partneriai gali turėti individualias duomenų bazes. Vykdomas duomenų replikavimas verslo partnerių duomenų bazėse. Kliento sistema atlieka komunikacinius veiksmus $Pr1_{sender}$ ir $Pr2_{receiver}$,o paslaugų sistema atlieka veiksmus $Pr1_{receiver}$ ir $Pr2_{sender}$ viso verslo transakcijos ciklo metu. Siunčiami pranešimų srautai yra asinchroniniai.

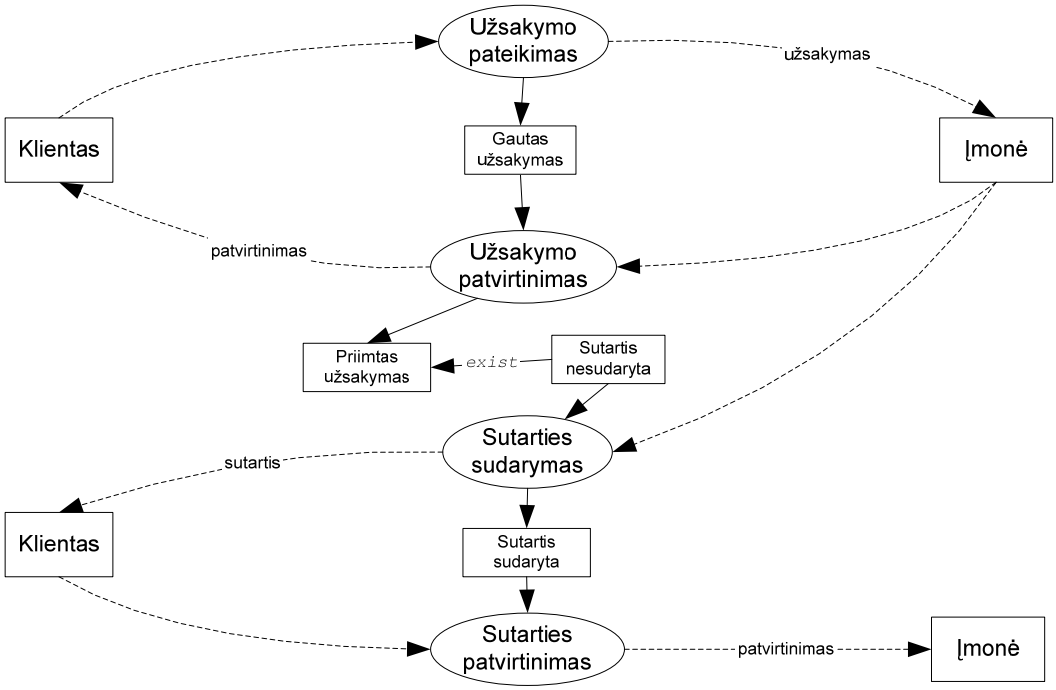


3 pav. Išplėstos komunikacinės veiksmų kilpos dekompozicija

Naudojant komunikacines veiksmų kilpas galima detaliai sumodeliuoti verslo procesų transakcijas vizualiai atvaizduojant veiksmus ir būsenų kaitą.

2.1.4. **Verslo transakcijų būsenos**

Veiksmų nuoseklumui apriboti naudojamos gyvavimo ciklo (*exist*) būsenų priklausomybės [18]. Būsenų gyvavimo ciklo priklausomybė reiškia, kad atitinkama (priklausomoji, vaikinė) būsena negali egzistuoti kol neegzistuoja tėvinė būsena (pavyzdžiui, negalima sudaryti sutarties užsakymui, kuris dar nėra priimtas). Gyvavimo ciklo būsenų priklausomybė iliustruojama *4 paveiksle*:



4 pav. Būsenų gyvavimo ciklo (exist) priklausomybė

Bendru atveju verslo transakcijos veiksmas turi *prieš* ir *po* būseną. Elementari binarinė kilpa susideda iš dviejų veiksmų, kurių kiekvienas turi *prieš* ir *po* būsenas [17], [19] (iš viso 4 būsenos). Pirmojo veiksmo *po* būsena gali sutapti su antrojo veiksmo *prieš* būsena (*4 pav.* būsena "*Gautas užsakymas*"). Belieka tik aprašyti būsenas nusakančių atributų aibes, kurios vizualiai nepateikiamos. Būsenas galima aprašyti naudojantis verslo transakcijų *BeginsWhen*, *EndsWhen*, *PreCondition* ir *PostCondition* elementais [3]:

- PreCondition – veiksmui išorinės būsenos, kuri reikalinga veiksmui pradėti, apibrėžimas.
- PostCondition – veiksmui išorinės būsenos, kuri reikalinga veiksmui pasibaigus, apibrėžimas (ši būsena neegzistuoja prieš veiksmo vykdymą, tačiau atsiranda pasibaigus veiksmui).
- BeginsWhen – veiksmui išorinio įvykio, kuris pradeda veiksmą, apibrėžimas (PreCondition + kiti kintamieji = BeginsWhen).
- EndsWhen – veiksmui išorinio įvykio, kuris užbaigia veiksmą, apibrėžimas (PostCondition + kiti kintamieji = EndsWhen).

Šiais elementais apibrėžiamos verslo taisyklės, kurios gali būti naudojamos ne vien tik anotacijai, bet ir paverčiamos į apskaičiuojamas išraiškas, kurias galima naudoti būsenų ir viso verslo proceso valdymui.

2.2. **Verslo transakcijų valdymas**

Vykstant verslo procesui viena po kitos yra vykdomos verslo procesą sudarančios transakcijos. Verslo proceso vykdymas nuo jo inicijavimo iki užbaigimo trunka priklausomai nuo laiko apribojimų, kurie taikomi verslo transakcijoms. Laiko apribojimai gali būti grindžiami verslo taisyklėmis arba tiesiog remiantis patirtimi bei logika, kai žinomas apytikris tam tikro veiksmo atlikimo laikas. Verslo transakcijos ar ją sudarančių veiksmų atlikimo laikas negali būti lygus nuliui.

Verslo transakcijų duomenys saugomi duomenų bazėse, todėl atsiranda poreikis susieti verslo transakciją su duomenų bazės transakcija. Priklausomai nuo duomenų bazių skaičiaus verslo transakciją atitinka dvi (kai yra viena DB) arba keturios (kai yra dvi DB) duomenų bazių transakcijos. Duomenų bazės transakciją gali sudaryti viena arba kelios DB užklausos.

2.2.1. **Verslo transakcijų klaidų valdymas**

Vykstant verslo transakcijai gaunamas sėkmės arba nepasisekimo rezultatas. Nepasisekimo rezultatas gali būti dvejopas:

- *Verslo transakcijos klaida*. Kaip jau buvo minėta, verslo transakcija yra darbo vienetas, tačiau ją sudaro eilė operacijų. Taigi, jeigu įvyksta klaida vykdant šias operacijas, tuomet reikia viską atstatyti į prieš tai buvusią būseną (*angl. rollback*). Vienai verslo transakcijai įvykdyti reikia keleto DB užklausų (skaitymo ir/arba rašymo į DB). Kai įvyksta klaida, labai svarbu užtikrinti tai, kad duomenų bazėje nebūtų likę dalies transakcijos įvykdytų užklausų rezultatų. Tarkime transakcijai įvykdyti reikia dviejų užklausų ir įvykus pirmajai atsiranda klaida, dėl kurios negalima sėkmingai užbaigti transakcijos. Tuomet reikia atstatyti DB pakeitimus, kuriuos padarė pirmoji užklausa. Pavyzdžiui, Saga modelyje [7] transakciją siūloma sudaryti iš elementarių transakcijų grandinės, kurioje kiekviena elementari grandinės transakcija turi ir savo kompensacinę transakciją. Jei elementari transakcija neįvykdoma teisingai, tai atstatoma ankstesnė būsena, o valdymas perduodamas atitinkamai kompensacinei transakcijai. Kompensacinę transakciją galima apibrėžti kaip veiksmus (DB užklausas), kurie semantiškai atstato verslo transakcijos padarytus dalinius pakeitimus nevykdant kaskadinio susijusių transakcijų nutraukimo, atstatant sistemą į pastovią (*angl. consistent*) būseną [8]. Po klaidos atstatymo galima pakartotinai vykdyti verslo transakciją (iš karto arba vėliau). Jeigu transakcija yra sėkmingai įvykdyta, tuomet galima pereiti prie kitos transakcijos vykdymo.

- *Verslo proceso nutraukimas*. Verslo proceso vykdymo metu jis vieno iš partnerių gali būti nutrauktas. Nutraukimo metu verslo proceso vykdymas jau gali būti pažengęs į priekį ir keletas verslo transakcijų jau gali būti įvykdyta. Modeliuojant verslo transakcijas turi būti apibrėžti veiksmai vykdomi proceso nutraukimo atveju. Pavyzdžiui, gali būti paliekama ir saugoma informacija apie nutrauktą verslo procesą arba gali būti vykdomas DB atstatymas (*angl. rollback*) panašus į ankščiau minėtą atstatymą, tik šiuo atveju reikės atstatyti iki proceso nutraukimo įvykdytas transakcijų užklausas.

Pirmasis klaidos tipas yra techninio pobūdžio, pavyzdžiui, nepavyko prisijungti prie duomenų bazės, o antrasis susijęs su verslo santykių būsena, pavyzdžiui, vienas iš partnerių nesilaikė įsipareigojimų, dėl to teko nutraukti vykstantį verslo procesą.

2.2.2. **Duomenų bazės atstatymas po klaidų**

Norint įvykdyti duomenų bazės atstatymą po nesėkmingai vykusio verslo proceso, reikia žinoti, kokie veiksmai buvo įvykdyti iki klaidos (nesėkmės) atsiradimo. Tokiu būdu yra būtina sekti vykstančių verslo transakcijų atliekamus veiksmus, t.y. duomenų bazės transakcijas. Turi būti saugojama atliktų DB užklausų istorija. Nutraukus verslo procesą peržiūrima to verslo proceso vykdymo metu atliktų užklausų istorija ir vykdomos kompensacinės užklausos, kurios atstato bei panaikina nereikalingus duomenų bazės įrašus. Būtina pabrėžti, kad DB atstatymas turi būti vykdomas atgaline tvarka, t.y. jeigu buvo įvykdytos dvi viena po kitos sekančios užklausos, tuomet atstatant DB pirmiausia turi būti įvykdyta antrąją užklausą atitinkanti kompensacinė užklausa, o tik po to pirmąją užklausą atitinkanti kompensacinė užklausa. Vietoj atliktų veiksmų saugojimo galima saugoti tik kompensacines užklausas, kurios sudaromos iškart vykstant verslo transakcijų DB užklausoms (tokioms kaip *Insert*, *Update* ir *Delete*). Pavyzdžiui, jeigu buvo įterpiamas įrašas į DB (panaudojant *Insert* užklausą), tuomet po įterpimo sudaroma šio įrašo pašalinimo kompensacinė užklausa (*Delete* užklausa) [8].

DB atstatymas iš pirmo žvilgsnio atrodo nesudėtingas, pavyzdžiui jeigu buvo pateiktas užsakymas, jį atšaukus ištrinama visa šio užsakymo informacija. Tačiau bendru atveju jeigu užsakymas buvo atšauktas ne iš karto po jo pateikimo, tuomet pateikto užsakymo informacija galėjo būti pakeista arba panaudota priimant sprendimus kitose verslo transakcijose. Pavyzdžiui, jeigu vakar pateiktas užsakymas sumažino turimų prekių sandėlyje skaičių nuo 100 iki 99, o šiandien prekių sandėlyje yra tik 37, tuomet ištrindami vakarykštį užsakymą ir atstatydami reikšmę atgal į 100 įvestume klaidą. Galima vietoj atstatymo į originalią reikšmę (į 100) atstatyti naudojant šiuo

metu esančių prekių sandėlyje skaičių (37) pridedant vienetą. Toks sprendimas yra geras tik tuo atveju, kai užsakymas atšaukiamas nedelsiant, tačiau jeigu prekė jau buvo paimta iš sandėlio ir paruošta išsiuntimui, tuomet užsakymo nutraukimas apima prekės perkėlimą atgal į sandėlį ir netgi pristatymo (pervežimo) įmonės perspėjimą, jog prekės nereikia pristatyti, jeigu ji jau buvo vežama klientui. Taigi yra dalykų, kurių duomenų bazės atstatymas taip paprastai negali padaryti. Taigi kompensavimas priklauso nuo laiko, kuomet jis vykdomas. Užsakymo panaikinimo praėjus kelioms minutėms po jo pateikimo procesas skiriasi nuo užsakymo panaikinimo praėjus savaitei, kai užsakymas jau įvykdytas ir prekės pristatytos. Be to gali būti transakcijų, kurių atstatymas nėra pageidaujamas. Pavyzdžiui, jeigu klientas pateikdamas užsakymą įvedė pristatymo adreso arba banko kortelės numerio pakeitimus, tuomet panaikinus užsakymą klientas gali norėti palikti adreso ir kortelės numerio pakeitimus neatstatant jų į duomenis, kurie buvo prieš užsakymo pateikimą. Bendru atveju norint užtikrinti saugų duomenų atstatymą, jį turi atlikti ta pati programinė įranga, kuri įvykdė verslo transakcijas, nes tik ji žino ką reikia kompensuoti kiekvienoje transakcijoje [9].

Galima išskirti tris užklausų tipus, kuriems reikalingos kompensacinės užklausos: *Insert*, *Update*, *Delete*. Užklausos plačiau aptariamos *2 lentelėje*:

2 lentelė. Kompensacinių užklausų formavimas

Įvykdytos užklausos tipas	Kompensacinė užklausa	Reikalingi duomenys	Pastaba
Insert	Delete	Reikia žinoti įterpto įrašo unikalų identifikatorių (*id*)	
Update	Update	Reikia žinoti atnaujinto įrašo(-ų) id ir duomenis, kurie buvo atnaujinti	Unikalūs identifikatoriai negali būti keičiami. Jie išlieka tokie patys nuo įrašo sukūrimo iki jo ištrynimo
Delete	Insert	Reikia turėti visą pašalinto įrašo informaciją.	Norint pagreitinti DB atstatymą galima neįterpinėti ištrintų įrašų Tokiu būdu nereikėtų vykdyti ir tų atstatomųjų užklausų, kurios yra susiję su ištrinta informacija

Prie kiekvieno duomenų bazėje saugomo verslo proceso duomenų įrašo galima pridėti po papildomą laukelį, kuris žymės ar įrašas buvo ištrintas. Tuomet jeigu atliekant verslo transakciją yra vykdoma informacijos ištrynimo (*Delete*) užklausa, informacija fiziškai nėra ištrinama iš DB, o tik įrašas papildomame laukelyje pažymimas kaip ištrintas. Kitaip sakant vykdomas ne fizinis, o loginis ištrynimas. Norint atstatyti tokiu būdu pašalintą įrašą tereikia tik nuimti ištrynimo žymę. Tuomet

verslo proceso informacijos ištrynimas ir ištrynimo kompensacinė užklausa bus vykdoma atliekant *Update* užklausą vietoje *Delete* ir *Insert* užklausų.

2.2.3. **Verslo transakcijos paskirstytose duomenų bazėse**

Aptartas klaidų valdymas sudėtingėja kai verslo partneriai turi atskiras duomenų bazes, kurios gali būti skirtinguose serveriuose. Tokiu atveju transakcija turi būti sinchronizuota tarp dviejų duomenų bazių, kurios yra skirtingose vietose. Pavyzdžiui, jeigu pervedame pinigus iš vienos sąskaitos į kitą, tuomet išėmus pinigus iš vienos sąskaitos svarbu, kad pinigai būtų pridėti į kitą sąskaitą. Sinchronizuojant verslo transakcijas tarp dviejų serverių atstatymas po klaidų vykdomas taip, kaip ir viename serveryje, tik tai atliekama dviejose duomenų bazėse.

Vykdant verslo transakcijas tarp dviejų duomenų bazių transakciją inicijuojantis serveris turi žinoti apie kitame serveryje esančios duomenų bazės lenteles. Tam naudojamas duomenų bazės replikavimas. Replikuojama ne visa verslo partnerių duomenų bazė, o tik jos dalis, kuri reikalinga verslo transakcijoms vykdyti.

Vykdant DB replikavimą galima naudotis vienu iš dviejų replikavimo metodų [20]:

- **Replikavimas pranešimų pagrindu**

  Dauguma komercinių replikavimo produktų naudoja replikavimą pranešimų pagrindu (pranešimų eilės, "išsaugojimo ir persiuntimo" replikavimas), kuris aptinka pakeitimus ir išsaugo pakeistų duomenų kopijas vienoje ar daugelyje pakeitimų istorijos eilių vėlesniam perdavimui. Kai kurie produktai naudoja duomenų bazės transakcijų žurnalus vietoje pakeitimų istorijos ar eilės. Pranešimų eilių paketai perduodami elektroniniu paštu, ftp, MQ (Message Queuing - pranešimų eilių produktais), ar panašiomis technologijomis.

  Replikavimas pranešimų pagrindu iš prigimties yra vienakryptis ir naudoja vieną išeinančią bei vieną įeinančią eilę šaltinyje ir paskirties serveryje atitinkamai. Dvikryptis replikavimas paprastai užtikrinamas naudojant antrą tokių eilių porą, tik priešinga kryptimi. Tokiu būdu naudojamos keturios eilės.

- **Replikavimas jungties pagrindu**

  Pagerėjus ryšių kokybei ir atsiradus galimybei užtikrinti pastovų ryšį, vis daugiau naudojamas replikavimas užmezgant tiesioginį ryšį tarp duomenų bazių. Tokiu būdu pakeitimų nebereikia saugoti žurnaluose ar formuoti jų eilių kiekvienai paskirties duomenų bazei sistemoje. Paskutiniai duomenys nuskaitomi tiesiogiai nuo duomenų bazės, konfliktai sprendžiami dinamiškai abiejose ryšio pusėse.

Pagrindinės problemos, kylančios replikavime pranešimų pagrindu, yra patikimumas, prognozuojamumas ir efektyvumas:

- *Neprognozuojamas saugojimo vietos poreikis*. Vietos poreikis pasikeitimų išsaugojimui žurnaluose ir pranešimuose yra kintantis ir neprognozuojamas. Žurnale arba eilėje turi būti visi pakeitimai skirti kiekvienam sistemos serveriui. Jei su bent vienu serveriu kontakto nebuvo savaitę, žurnale arba eilėje turi būti visi tos savaitės pakeitimai, net jei ir su kitais serveriais apsikeitimas vyksta kas valandą.
- *Įgimtas pažeidžiamumas ir poreikis administravimui*. Žurnalų ar eilių saugojimo vietos limitų viršijimas sąlygoja duomenų praradimą, nes nelieka pakankamai informacijos pilnai atnaujinti paskirties serverį. Tai taip pat reikalauja administratoriaus įsikišimo duomenų bazių atstatymui, kas paprastai būna pažeistos duomenų bazės pakeitimas pagrindinės duomenų bazės (šaltinio) pilna arba daline kopija.
- *Neefektyvus duomenų dubliavimas*. Žurnaluose ir eilėse saugomi pakeisti duomenys, o kiekvieno įrašo reikšmių papildomų kopijų saugojimas yra neefektyvus. Taip pat dauguma tokių sprendimų saugo visus pakeitimus atliktus su kiekvienu įrašu, o ne tik paskutinius pakeitimus. Tokiu būdu, jei įrašas keičiamas keletą kartų tarp replikavimo sesijų, šie metodai saugo ir perduoda keletą skirtingų to paties įrašo kopijų.
- *Įgimti rizikos taškai ir prastas išplečiamumas*. Pranešimais grįstas replikavimas sistemą skirsto į šaltinius ir paskirties taškus arba "šeimininkus" ir "tarnus". Kiekvienoje tokioje poroje yra keletas rizikos taškų, kadangi procesas susideda iš eilės tarpusavyje susijusių procesų ir eilių (pvz., išrašymas, išeinančių paketų formavimas, perdavimas, paketų priėmimas ir įdiegimas). Tai iš esmės stebulės ir stipinų modelis, kuriame yra vienas centrinis rizikos taškas ir kurio išplečiamumą riboja tai, kiek serverių gali aptarnauti centrinis serveris.

Replikavimas jungties pagrindu lyginant su replikavimu pranešimų pagrindu turi tokius privalumus:

- *Nedideli ir prognozuojami reikalavimai papildomai vietai*. Kiekvienam replikuojamam įrašui yra saugomas apibrėžtas kiekis kontrolinės informacijos. Nėra kintamo dydžio pranešimų eilių, kurios auga.
- *Naudojimo paprastumas, patikimumas ir išplečiamumas*. Jungtimis grįstą replikavimą yra pigiau eksploatuoti, nes jis yra mažiau sudėtingas vartotojams ir administratoriams, turi mažiau rizikos taškų ir lengviau gali būti automatizuojamas. Užtikrinamas pilnai dvipusis replikavimas "taško su tašku" (peer-to-peer), kurio metu galima dinamiškai aptikti

replikavimo nesėkmes ir nukreipti duomenis kitur, kai kurie nors taškai yra nepasiekiami. Nebelieka apribojimų, kuriuos sąlygoja stebulės ir stipinų konfigūracijos. Vietoje keleto jungiamųjų taškų tarp bendradarbiaujančių procesų ir eilių, šiuo atveju naudojamas tik vienas taškas, vadinamas prisijungimu (jungtimi) arba sesija.

- *Savalaikis ir greitas konflikto aptikimas bei sprendimas*. Konfliktai aptinkami ir išsprendžiami vienu metu abiejose replikavimo sesijos dalyvių duomenų bazėse.

- *Efektyvus duomenų perdavimas*. Net jei tie patys duomenys yra keičiami daugelį kartų tarp replikavimo sesijų, perduodama tik paskutinė būsena, t.y. šių pakeitimų galutinis rezultatas.

2.3. **Verslo transakcijų modeliavimo metodų, standartų ir produktų apžvalga**

2.3.1. **Elektroninių ryšių valdymas ebXML**

Yra vykdoma ir keletas komercinių projektų, susijusių su verslo transakcijų modeliavimu. Vienas iš jų yra ebXML (*electronic business XML*) [10]. EbXML plėtoja globalios elektroninės rinkos idėją. Globali elektroninė rinka – tai vieta, kur bet kokio dydžio ir bet kurioje pasaulio vietoje esančios įmonės gali susitikti viena su kita ir tvarkyti verslo reikalus. Tai didelis projektas, kuris buvo inicijuotas OASIS (Organization for the Advancement of Structured Information Standards – struktūrizuotų Informacinių Standartų Pažangos Organizacija) bei UN/CEFACT (United Nations Center for Trade Facilitation and Electronic Business) ir buvo pristatytas 1999m. 2004m. kovo mėnesį. EbXML verslo procesą apibrėžia kaip verslo bendradarbiavimą, t. y. dviejų ar daugiau partnerių veiklą siekiant gauti tam tikrą rezultatą.

EbXML sudaryta iš keturių pagrindinių architektūrinių komponentų [11], [12], [13]:

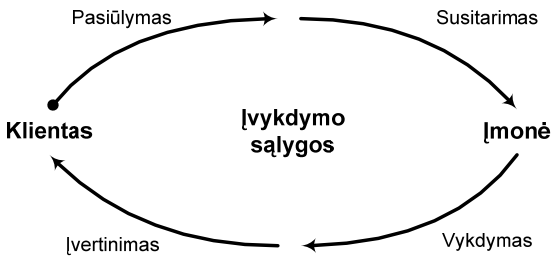
- Pranešimų siuntimo paslauga – tai standartinis būdas apsikeisti verslo pranešimais tarp organizacijų. Siunčiamas pranešimas gali būti ne tik XML formato, bet ir kitoks dokumentas, kuris bus perduotas patikimai ir saugiai.
- Registras – tai duomenų bazė, kurioje saugomi elementai palaikantys elektroninį verslą. Techniškai šnekant registras saugo informaciją apie elementus, kurie yra saugykloje (*repository*). Kartu registras ir saugykla yra duomenų bazė. Elementai saugykloje sukuriami, ištrinami bei atnaujinami per registro užklausas. Konkreti registro/saugyklos duomenų bazės realizacija nėra specifikuota. Yra žinoma tik tai, kaip kitos programinės priemonės bendrauja su registru ir minimalus informacijos modelis (kokio tipo informacija yra saugoma saugykloje), kurį palaiko registras. Registro elementais gali būti XML biznio dokumentų schemos, komponentų, naudojamų verslo procesų modeliavimui, apibrėžimai, prekybos partnerių sutartys.

- Prekybos partnerio informacija – bendradarbiavimo protokolo profilis (*Collaboration Protocol Profile* arba *CPP*) aprašo XML schemą, kuri detalizuoja tai, kaip organizacija elektroniniu būdu gali tvarkyti verslo reikalus. Ji nurodo tokius elementus, kaip organizacijos kontaktai ar kita organizacijos informacija, naudojami tinklo ir failų siuntimo protokolų tipai, tinklo adresai, saugumo realizacija, ir tai, kaip organizacija tvarko verslo reikalus. Bendradarbiavimo protokolo susitarimas (*Collaboration Protocol Agreement* arba *CPA*) aprašo tai, kaip dvi organizacijos susitarė elektroniniu būdu tvarkyti verslo reikalus. Tai suformuojama apjungiant dviejų organizacijų CPP. CPA gali būti naudojamas programinės įrangos, kuri atliks konfigūracijas leisiančias elektroniniu būdu tvarkyti verslo reikalus su kita organizacija. Tačiau nėra jokio konkretaus algoritmo, kuris nurodytų, kaip apjungti du CPP į vieną CPA.
- Verslo proceso specifikacijos schema (Business Process Specification Schema arba BPSS) – specifikacijos schema aprašo XML dokumentą, kuris apibūdina, kaip organizacija tvarko savo verslo reikalus. Taigi CPA/CPP aprašo techninius verslo reikalų tvarkymo elektroniniu būdu aspektus, o specifikacijos schema aprašo konkrečius verslo procesus. Specifikacijos schema nusako visą verslo procesą, nustato roles, operacijas, naudojamus verslo dokumentus, dokumentų srautus, teisinius aspektus, saugumo aspektus, verslo lygius ir būsenas. Specifikacijos schema gali būti naudojama programinės įrangos, kuri atliks konfigūracijas, norint elektroniniu būdu tvarkyti verslo reikalus su kita organizacija.

Svarbu suprasti kad ebXML nėra viena technologija, tai greičiau specifikacijų rinkinys skirtas kurti e-verslo struktūrą. Kaip matome iš architektūrinių komponentų aprašymo, bendravimui tarp organizacijų naudojama ebXML pranešimų siuntimo paslauga. Verslo pranešimuose naudojami informaciniai komponentai, kurie surašyti komponentų kataloge [14]. Siunčiamuose verslo dokumentuose atpažįstami šie komponentai, ir iš jų tampa aiškus pranešimo kontekstas. Galima sukurti verslo procesų redagavimo įrankį, kurio pagalba specifikacijos schemos gali būti generuojamos automatiškai surenkant verslo proceso duomenis iš vartotojo.

2.3.2. **Winograd-Flores komunikacinių kilpų modelis**

Autorių Terry Winograd ir Fernando Flores pasiūlytas verslo bendradarbiavimo modelis bendravimą vaizduoja uždara darbų sekų kilpa (*5 paveikslas*) ir yra panašiausias į EM (*Enterprise Modeling*) metodą.



5 pav. Winograd-Flores darbų sekų kilpa

5 paveiksle pavaizduota veiksmo, kurį įmonė sutinka įvykdyti tam, kad klientas gautų tam tikrą pasitenkinimą (*satisfaction*), darbų sekų kilpa. Kilpa įvykdoma keturiomis fazėmis [21], [22]:

1) *Pasiūlymas (pasiruošimas)*

   Klientas prašo (arba įmonė pasiūlo) tam tikro veiksmo įvykdymo pagal nustatytas sąlygas.

2) *Susitarimas (analizė)*

   Abi šalys prieina abipusio susitarimo dėl veiksmo vykdymo sąlygų įskaitant laikus, kuomet bus pradedami vykdyti kiti žingsniai.

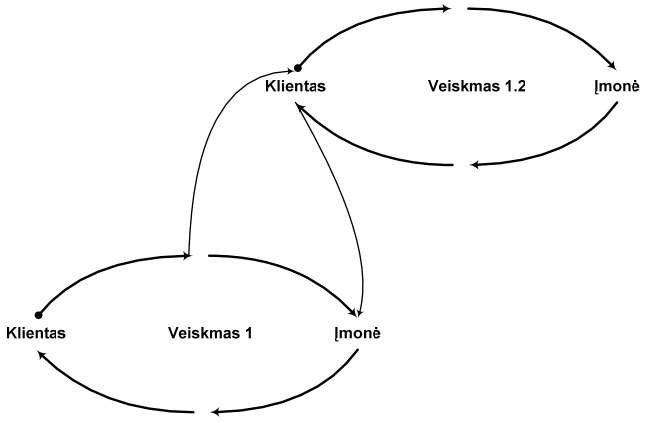
3) *Vykdymas*

   Įmonė įvykdo prašymą (veiksmą) ir praneša klientui apie užbaigimą.

4) *Įvertinimas*

   Klientas įvertina atliktą darbą ir pareiškia pasitenkinimą arba nurodo dalykus, kuriuos dar reikia padaryti, kad veiksmo vykdymas būtų užbaigtas.

Bet kuri fazė gali būti sudaryta iš kitos darbų sekų kilpos (*6 paveikslas*), kurios tikslas gali būti detalesni paaiškinimai, tolesnės derybos dėl sąlygų, dalyvių įsipareigojimų pasikeitimas ir kt. Taigi modelyje vykdomos užduotys apibrėžiamos per reikalavimus ir įsipareigojimus išreikštus kilpomis.



6 pav. Winograd-Flores darbų sekų kilpos detalizavimas

Winograd-Flores modelio pagrindinis privalumas yra tai, kad nei vienas verslo proceso žingsnis (kilpa) nėra traktuojamas kaip užbaigtas, kol atitinkamo žingsnio nepatvirtina klientas. Pagal šį modelį verslo bendradarbiavimas sudarytas iš aibės kilpų, kurios vaizduoja bendravimą tarp šalių ir yra įtraukiamos skirtinguose proceso lygiuose [22]. Kuomet verslo proceso žingsnis teisingai užbaigiamas, tuomet kilpa užsidaro ir pradedamas vykdyti kitas žingsnis.

2.3.3. **BTP protokolas**

BTP [15] – tai XML pagrindu sudarytas protokolas skirtas vykdyti transakcijas internetu. Naudojamas dviejų fazių patvirtinimo protokolas darbų sekoms kontroliuoti. BTP sukurtas taikomųjų programų koordinavimui, kurios priklauso dviems ar daugiau verslo partneriams ir verslo transakcija apibrėžiama kaip nuosekli šių partnerių verslo būsenų kaita. Šis protokolas apibrėžia aktorius ir ryšius tarp jų. Yra du pagrindiniai ryšiai: valdymo ir aukštesnio – žemesnio (*angl. inferior - superior*). Valdymo ryšys – tai ryšys tarp elemento, kuris nustato transakcijos pabaigą, ir aktoriaus, kuris yra transakcijos medžio viršuje ir kuris sprendžia ar dalyvis patvirtina ar atšaukia transakciją. Aukštesnio – žemesnio ryšys - tai ryšys tarp transakcijų medžio elementų, kur aukštesnis elementas (koordinatorius) informuoja žemesnį elementą (dalyvį) apie sprendimą dėl transakcijos patvirtinimo ar atšaukimo. BTP yra laikomas vienu perspektyviausių projektų internetinių transakcijų standartų srityje. Jis atkreipia dėmesį i dažniausias internetinių transakcijų problemas: ilgos trukmės, resursų blokavimas, nepatikimas ryšys ir pan. Tačiau BTP specifikuoja tik pranešimus tarp transakcijų partijų ir todėl prireikia nemažai išteklių tolimesniems įdiegimo veiksmams. Kiekvienas transakcijos dalyvis turi specifikuoti savąjį, sudėtingą darbų sekos modelį bei komunikacinį protokolą.Vienas iš komercinių produktų sukurtų BTP pagrindu yra Hewlett Packard firmos Web Services Transaction (*HP-WST*).

2.3.4. **Elektroninio verslo modelių sudarymo kalba XLANG**

Microsoft Biz Talk serveryje naudojama vietinės reikšmės kalba XLANG [16]. Ji skirta formaliam verslo procesų specifikavimui. XLANG kiekvieno aktoriaus servisų sąsajų specifikavimui naudoja WSDL kalbą (*angl. Web Services Description Language*). Operacijų duomenų srautai nespecifikuodami. XLANG tikslas yra galimybė formaliai specifikuoti ilgos trukmės kompensuojamas transakcijas. Kompensacijos taikomos klaidų apdorojimui. XLANG siūlo lanksčius įrankius klaidų apdorojimo ir būsenų atstatymo uždaviniams. Transakcijas galima rinkti į rinkinius.

Verslo procesai apibūdinami kaip duomenų apsikeitimai tarp aktorių (paslaugų sistemų). Darbų seka apibrėžiama kaip blokinė struktūra. Transakcijų struktūros aprašomos transakcijų blokais, kuriuos sudaro bet koks skaičius transakcijų. Su kiekvienu tokiu bloku galima susieti ir kompensacinę transakciją. Jei įvyksta klaida, vykdomos kompensacinės transakcijos, kurioms galima nustatyti vykdymo tvarką, tačiau pagal nutylėjimą jos vykdomos atvirkštine tvarka.

Darbe [1] atlikus verslo transakcijų modelių lyginamąją analizę buvo prieita išvados, kad verslo transakcija skirtingai suprantama įvairiuose modeliuose ir neturi griežto formalaus apibrėžimo. Šiuolaikines verslo procesų modeliavimo metodų tendencijas geriausiai atspindi *ebXML* veiklos procesų modeliavimo metodologija.

2.4. **Verslo transakcijų modeliavimo ir valdymo išvados**

Išanalizavę keletą verslo transakcijų modeliavimo metodų ir standartų galime teigti, kad nėra vienareikšmiško elementarios verslo transakcijos apibrėžimo. Galima išskirti du panašius modelius, kurie yra tinkamiausi verslo transakcijų modeliavimui: Winograd - Flores darbų sekų modelis ir EM metodo komunikacinės veiksmų kilpos. Pirmasis modelis leidžia vizualiai pateikti veiksmų (darbų) sekas, o komunikacinės kilpos be veiksmų sekų pateikia ir būsenų kaitą.

Darbe pasiūlyta sudėtines verslo transakcijas (verslo procesus) modeliuoti binarinėm komunikacinėm veiksmų kilpom, o veiksmų nuoseklumui specifikuoti panaudoti gyvavimo ciklo (*exist*) priklausomybes. Binarinės kilpos įgyja veiksmų autonomiškumą, o veiksmų nuoseklumą galima kontroliuoti duomenų bazėse, kuriose yra saugomos verslo objektų būsenos. Binarinės veiksmų kilpos yra verslo transakcijų dinamikos šaltinis, iš kurių patogu realizuoti duomenų bazių transakcijas.

Pasiūlyta naudoti jungtimis grįstą replikavimą. Tai padidina replikavimo sistemos lankstumą, leidžia kaupti mažiau duomenų, sistemą galima sėkmingai plėsti.

3. DUOMENŲ BAZIŲ TRANSAKCIJŲ MODELIAVIMAS NAUDOJANTIS KOMUNIKACINĖMIS VEIKSMŲ KILPOMIS

Verslo transakcijas modeliuosime komunikacinėmis binarinėmis veiksmų kilpomis, kurios vizualiai pateikia tiek elementarių, tiek sudėtinių verslo transakcijų vyksmą. Binarinėmis kilpomis yra užduodami funkciniai reikalavimai. Bendru atveju naudojantis kilpomis sudaroma duomenų struktūra. Duomenų bazės kūrimas naudojantis binarinėmis veiksmų kilpomis darbe neanalizuojamas ir detaliai neprašomas. Darbe daroma prielaida, kad duomenų struktūros jau yra sudarytos ir mes modeliuojame verslo transakcijas, kurios palaiko esamą DB struktūrą, t.y. verslo transakcijos yra pritaikomos esamai DB.

3.1. Komunikacinių veiksmų kilpų sudarymas

Komunikacinė kilpa - būdas formaliai aprašyti verslo transakcijas tarp verslo partnerių. Komunikacinėmis veiksmų kilpomis galima pavaizduoti trijų tipų priklausomybes: statines, dinamines bei pragmatines. Tuo tarpu UML neturi pragmatinių priklausomybių ir statikai bei dinamikai naudojamos atskiros diagramos. Taigi komunikacinės veiksmų kilpos viename modelyje perteikia verslo transakcijos informaciją, kuriai perteikti naudojant UML kalbą prireiktų bent dviejų diagramų.

Modeliuojant komunikacines kilpas naudojami tokie elementai:

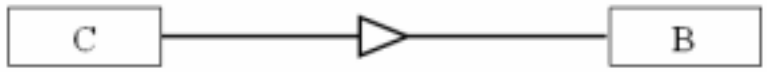
- vaidmenys – aktoriai, kurie tarpusavyje bendrauja,
- srautai – informacijos, duomenų srautai, kuriais keičiasi bendradarbiaujantys aktoriai,
- veiksmai – tikslinga veikla siekiant tam tikro rezultato,
- būsenos – parametrų rinkiniai nusakantys transakcijos vykdymo būklę bei kontroliuojantys veiksmą.

Jungiant elementus tarpusavyje sudaromas verslo transakcijos modelis.

3.1.1. Statinės priklausomybės

Statinės priklausomybės naudojamos modeliuojant verslo transakcijos būsenas ir jų tarpusavio priklausomybes. Galima išskirti šiuos statinių priklausomybių tipus (*3 lentelė*):

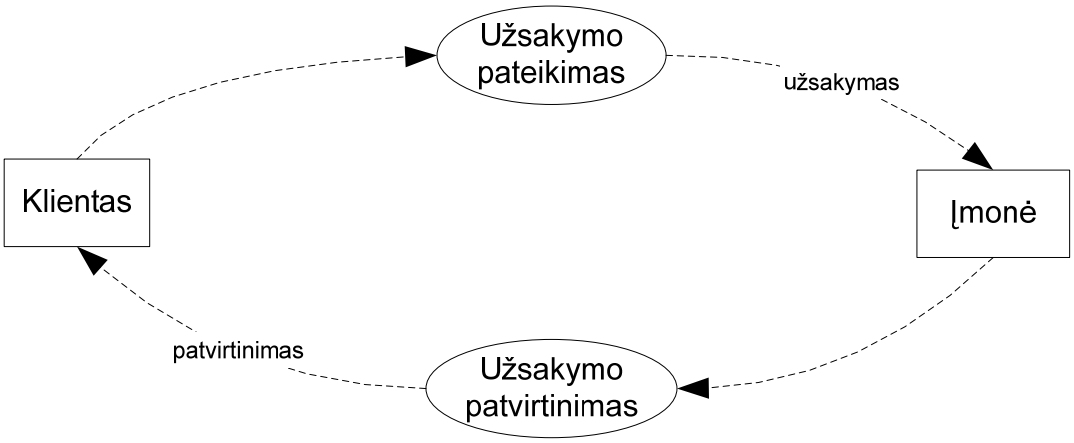
3 lentelė. Statinės priklausomybės

Statinės priklausomybės vaizdavimas	Kardinalumas
	(0..1; ?..?)
	(1..1; ?..?)
	(0..*; ?..?)
	(1..*; ?..?)
	C yra B komponentas
	B agregatas sudarytas iš C
	C paveldi B
	C yra B egzempliorius
	C ir B nesikerta, C ir B paveldi D

Statinėmis priklausomybėmis susietos būsenos dinaminėmis priklausomybėmis sujungiamos su kitais modeliavimo elementais.

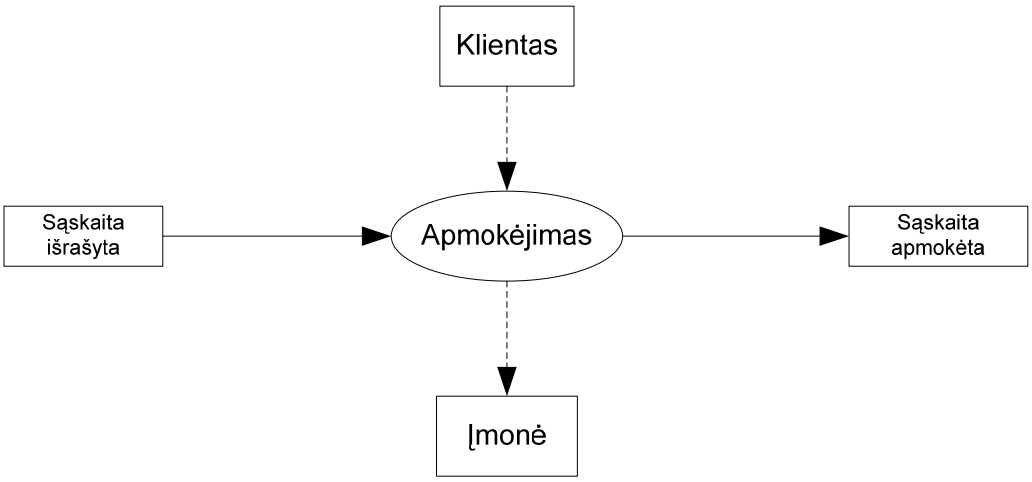
3.1.2. **Dinaminės priklausomybės**

Dinaminės priklausomybės atspindi verslo transakcijos dinamiką, verslo transakcijos vykdymo tvarką. Modeliuojant dinamiką vaidmenys ir veiksmai yra sujungiami srautais taip sudarydami komunikacinę kilpą (*7 paveikslas*). Kadangi veiksmus galima sutapatinti su panaudojimo atvejais duodančiais tam tikrą rezultatą veiksmą inicijuojančiam aktoriui, tuomet kilpų kaip ir panaudojimų atvejų modeliavimas vizualiai perteikia atitinkamos dalykinės srities funkcinius reikalavimus.



7 pav. Komunikacinė veiksmų kilpa

Vykstant komunikacinės kilpos veiksmams nuolat kinta verslo proceso būsena. Komunikacinės kilpos veiksmai atlieka būsenų pasikeitimą (*8 paveikslas*).



8 pav. Būsenų kaita komunikacinėje veiksmų kilpoje

Pranešimų srautai atitinka įvykius, kurie pažymi būsenos pasikeitimą. Ryšys tarp veiksmo ir įvykio pavaizduotas *9 paveiksle*.

9 pav. Ryšys tarp veiksmo ir įvykio

Kai kalbama apie veiksmus, tai dėmesys kreipiamas į veiksmų seką, skirtą pakeitimams atlikti, o kai kalbama apie įvykius, tai dėmesys kreipiamas į pasikeitimų rezultatus.

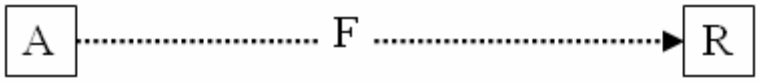
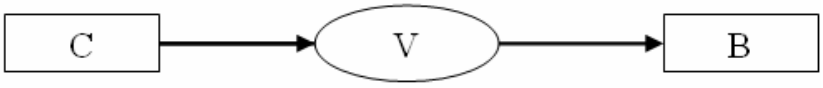
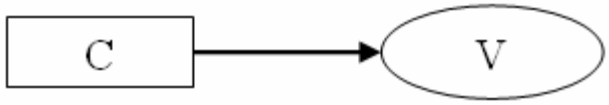
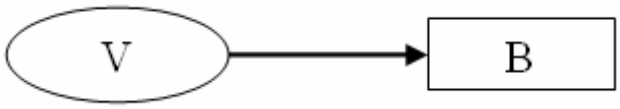
Nereikalaujama, kad procesai visada baigtųsi įvykiais. Bet tik tie procesai, kurie pakeičia objektų būsenas, gali būti interpretuojami kaip veiksmai. Būsena kontroliuoja veiksmą, kuris tęsiasi tol, kol jo nenutraukia įvykis, fiksuojantis perėjimą į kitą būseną (*10 paveikslas*).

10 pav. Būsenų kaita

Tam, kad veiksmas būtų korektiškai atliktas, turi būti korektiškai specifikuoti pradinės ir kitos būsenos apribojimai. Būsenų apribojimai vaidina lemiamą vaidmenį atliekant veiksmus, nes būsenos juos kontroliuoja. Pradinė būsena specifikuoja apribojimus, kurie turi būti patenkinti, norint inicijuoti veiksmą (atitinka *prieš sąlygą*). Kita būsena specifikuoja apribojimus, kurie turi būti patenkinti užbaigus veiksmą (atitinka *po sąlygą*).

Bendru atveju gali būti tokios dinaminės priklausomybės (*4 lentelė*):

4 lentelė. Dinaminės priklausomybės

Dinaminės priklausomybės vaizdavimas	Reikšmė
	Komunikacinis srautas F einantis iš aktoriaus A į aktorių R
	Komunikacinis veiksmas V inicijuojamas aktoriaus A ir nukreiptas į aktorių R
	Perėjimas iš būsenos C į būseną B įvykus veiksmui V
	Objektas sunaikinamas būsenoje C
	Objektas sukuriamas būsenoje B

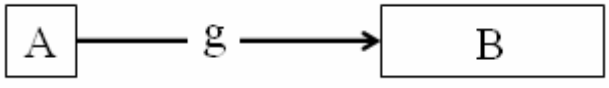
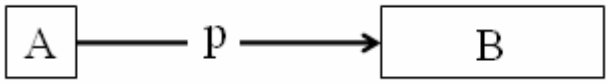
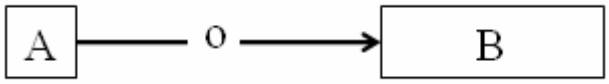
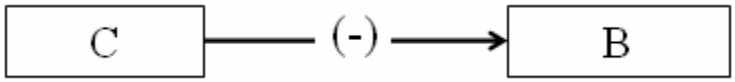
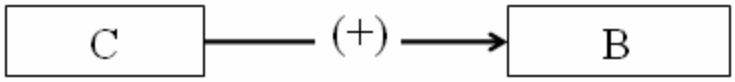
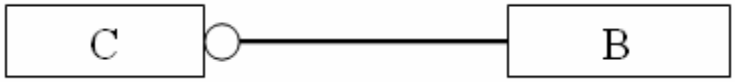
Komunikacinių kilpų elementus sujungę statinėmis bei dinaminėmis priklausomybėmis gauname formaliai specifikuotą verslo procesą.

3.1.3. **Pragmatinės priklausomybės**

Statinės ir dinaminės priklausomybės komunikacinėse kilpose atsako į klausimus *kas*, *ką*, *kada* ir *kur* turi vykdyti Tuo tarpu pragmatinės priklausomybės atsako į klausimą *kodėl* nurodydamos transakcijų vykdymo motyvaciją.

Išskiriamos tokios pragmatinės priklausomybės (*5 lentelė*):

5 lentelė. Pragmatinės priklausomybės

Pragmatinės priklausomybės vaizdavimas	Reikšmė
	Aktorius A turi tikslą B
	Aktorius A turi problemą B
	Aktorius A turi galimybę B
	Objektas būsenoje C neigiamai veikia objektą būsenoje B
	Objektas būsenoje C teigiamai veikia objektą būsenoje B
	C patikslina B

Statinės, dinaminės bei pragmatinės priklausomybės leidžia sukurti modelį užtikrinantį kai kuriuos semantinio integralumo kriterijus:

- Darna (angl. *coherence*) – galinės būsenos suderinamos su aktorių tikslais.
- Pilnumas (angl. *completeness*) – nėra probleminių būsenų, kurios rodytų būtinybę tobulinti kompiuterizuotą sistemą.
- Įgyvendinamumas (angl. *viability*) – komunikacinė kilpa įgyvendinama (*1 paveikslas*), jei $A1 \xrightarrow[p]{} B11, A2 \xrightarrow[o]{} B22, A2 \xrightarrow[p]{} B21, A1 \xrightarrow[g]{} B12,$

  tuomet $B11 \xrightarrow[-]{} B12, B22 \xrightarrow[+]{} B12, B22 \xrightarrow[-]{} B11, B12 \xrightarrow[-]{} B21;$

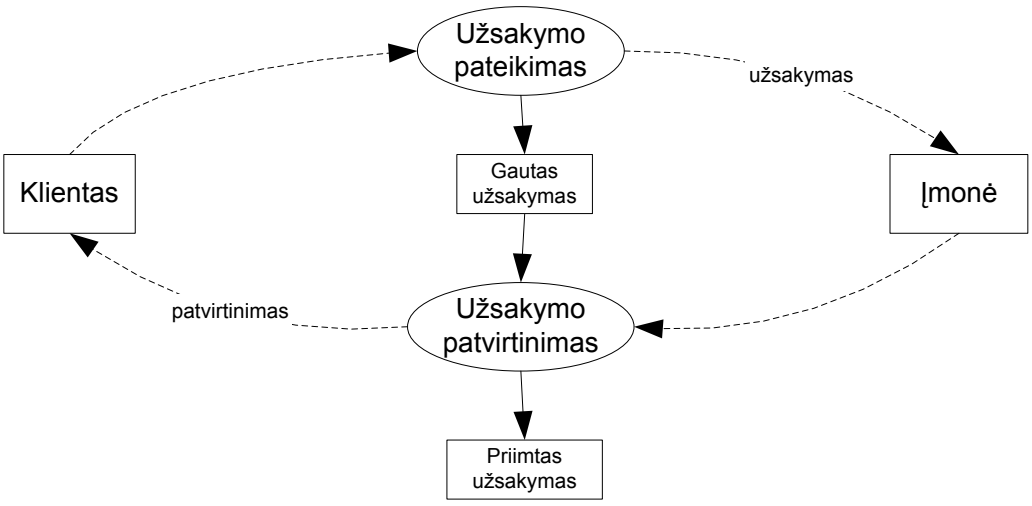
  Kitaip tariant, komunikacinė kilpa yra įgyvendinama, jei atsakovo problema neigiamai veikia iniciatoriaus problemą ir atsakovas gali įvykdyti reakcijos veiksmą, kurio rezultatas teigiamai veikia tiek atsakovo, tiek iniciatoriaus tikslus.

Esant didelėms sudėtinėms komunikacinėms kilpoms naudojantis pragmatinėmis priklausomybėmis galima surasti verslo proceso sėkmingo įgyvendinimo kelią, t.y. kelią, kuriuo turi vykti transakcijų vykdymas norint pasiekti pageidaujamą rezultatą (teigiamos (+) priklausomybės yra sujungtos su tiksline būsena).

3.2. Verslo transakcijų užduotų komunikacinėmis veiksmų kilpomis parametrų parinkimas

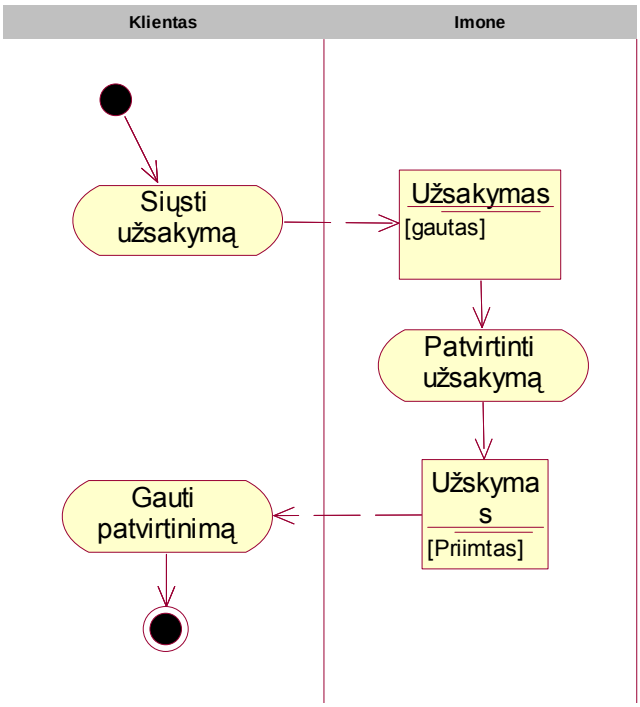
Atlikus verslo transakcijų modeliavimą gaunamas detaliai specifikuotas verslo proceso modelis. Gautas modelis gali būti panaudotas informacinės sistemos, vykdančios sumodeliuotas transakcijas, kūrimui. Tačiau norint kompiuterizuoti verslo transakcijas, būtina komunikacines veiksmų kilpas detaliai išanalizuoti identifikuojant su transakcija susijusių aktorių, veiksmų, informacijos srautų bei būsenų informaciją, kuri reikalinga sistemai funkcionuoti. Pereinant prie kito verslo transakcijų projektavimo etapo yra dekomponuojami verslo transakcijų elementai iki žemiausio lygio, kuriame yra išvardijami verslo transakcijas nusakantys parametrai. Kitaip sakant vykdomas transakcijų parametrų parinkimas. Parametras saugo verslo informaciją. Vykstant verslo transakcijoms verslo partneriai keičiasi informacija, kuri aprašoma parametrų rinkiniais. Turėdami aiškiai apibrėžtą informaciją galime ją saugoti duomenų bazėje, nes parametrai nusako duomenų struktūrą reikalingą tai informacijai išsaugoti.

Nustatant verslo proceso parametrus pirmiausia reikalingas detalus verslo proceso modelis. Modelis turi būti pakankamas reikalingos verslo informacijos identifikavimui, pavyzdžiui, turime modelį, kurio dalis (*11 paveikslas*) vaizduoja pirkėjo reikalavimų siuntimą norimai paslaugai gauti.



11 pav. Komunikacinės veiksmų kilpos pavyzdys

Norint parodyti komunikacinių kilpų atitikmenį UML kalboje, *12 paveiksle* pavaizduota kilpa perteikiama UML veiklos procesų diagrama, kuri vaizduoja veiksmus, srautus bei būsenas.

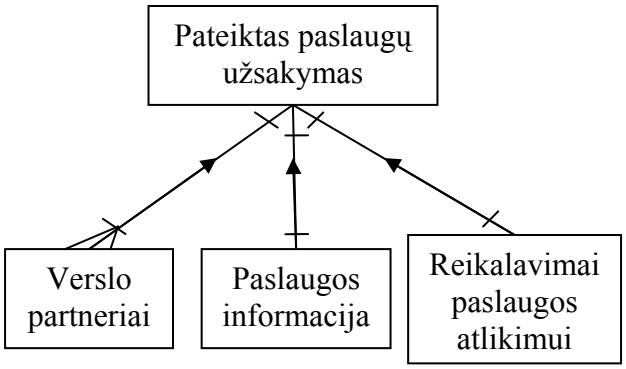


12 pav. Veiklos procesų diagramos pavyzdys

Naudojantis *11 paveiksle* pavaizduota kilpa galima išskirti reikalingą verslo informaciją. Bendru atveju galima išskirti trijų tipų verslo transakcijų duomenų komponentus: srautai, prieš ir po būsenos, verslo partnerius identifikuojantys duomenų komponentai.

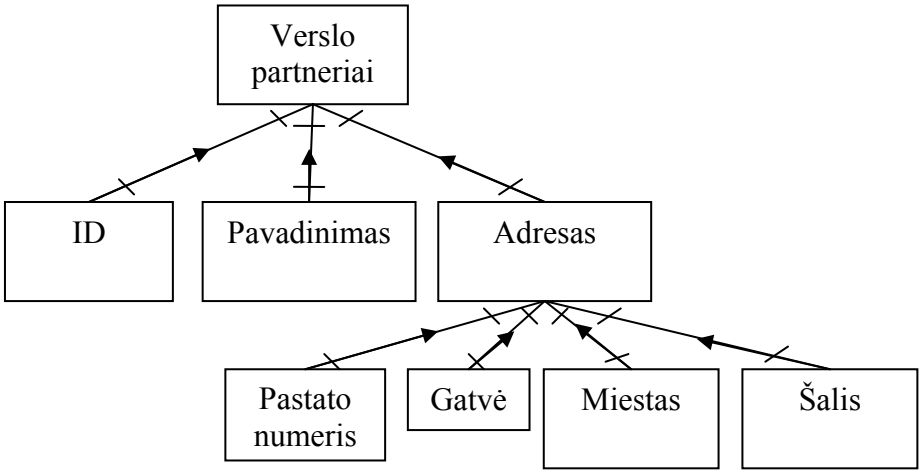
Nustatant parametrus, galima naudotis tokiais žingsniais:

- Padalinti verslo informaciją į komponentines grupes (*13 paveikslas*) ir joms suteikti pavadinimus. Komponentinėms grupėms vaizduoti panaudojamos statinės priklausomybės. Pavyzdžiui:



13 pav. Komponentinių informacijos grupių pavyzdys

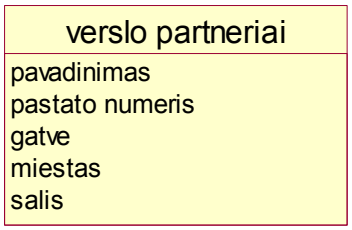
- Kiekvieną informacijos komponentinę grupę dalinti į mažesnius komponentinius vienetus (*14 paveikslas*). Kiekviena komponentinė grupė gali turėti identifikatorių ir kitas charakteristikas. Pavyzdžiui:



14 pav. Informacijos komponentinės struktūros pavyzdys

- Tęsti komponentinį dalijimą toliau kol pasiekiamas žemiausias lygis, kuomet informacija jau negali būti padalijama.
- Komponentiniai informacijos vienetai žemiausiame lygyje atitinka parametrus, kurie naudojami vykstant verslo procesui.

Gautas parametrų rinkinys susijęs su tam tikru objektu gali būti pavaizduotas UML kalboje kaip objekto klasė. Pavyzdžiui, *15 paveiksle* pavaizduota *verslo partnerių* klasė (klasėje ID parametras bei parametras turintis komponentų (*Adresas*) nevaizduojamas).

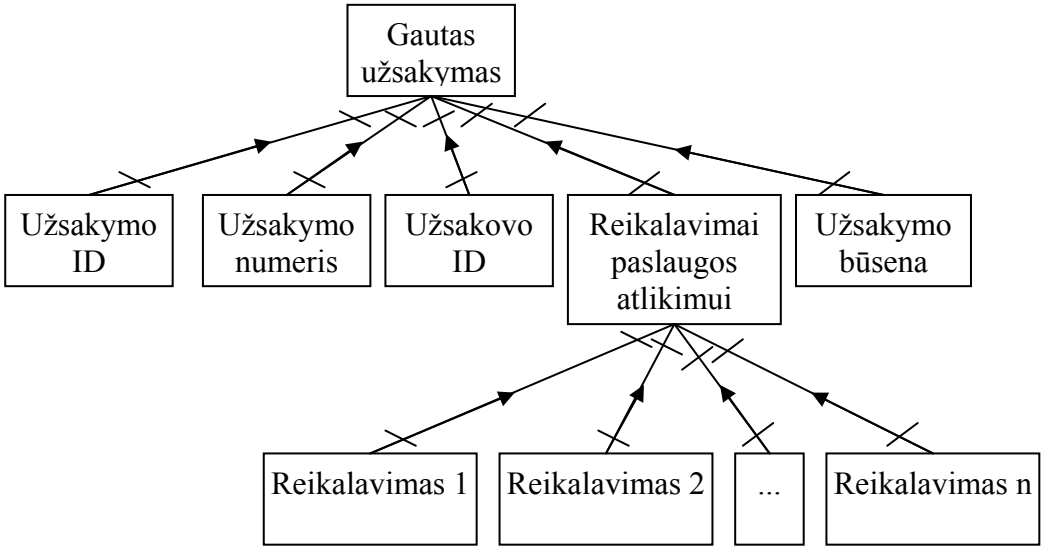


15 pav. UML klasės pavyzdys

Pastabos nustatant parametrus:

- Kiekviena verslo transakcija turi savo ID parametrą.
- Būtina apibrėžti identifikuojančius parametrus tokius kaip ID arba vardas.
- Nustatant informaciją susijusią su įvykiais reikia nurodyti datos/laiko parametrą ir vietos, su kuria susijęs veiksmas, parametrą.

Parametrų rinkiniai naudojami ne tik srautų informacijos, bet ir būsenų aprašymui. Pavyzdžiui, *16 paveiksle* pateiktas būseną *gautas užsakymas* nusakančių parametrų rinkinio fragmentas.



16 pav. Būsena nusakančių parametrų rinkinio fragmentas

Vykstant transakcijai parametrų kiekis ir jų reikšmės keičiasi. Esant tokiems pasikeitimams keičiasi ir verslo proceso būsena. Saugant būsenas nusakančių parametrų reikšmes yra valdomas verslo procesas. Gyvavimo ciklo (*exist*) būsenų priklausomybė realizuojama susietų būsenų bendrais parametrų rinkiniais. Pavyzdžiui (*4 paveikslas*), negali būti sudarinėjama sutartis užsakymui, kuris nebuvo priimtas. Tai reiškia, kad prieš sudarant sutartį užsakymui su tam tikru užsakymo numeriu, turi egzistuoti priimtas užsakymas, kuris turi nustatytą užsakymo numerio parametrą su atitinkama reikšme. Taigi užsakymo numerį galima traktuoti kaip *exist* priklausomybės parametrą, siejantį dvi atskiras būsenas. Bendru atveju gyvavimo ciklo priklausomybės parametrai yra šia priklausomybe susietos tėvinės būsenos parametrų poaibis.

Galima pastebėti, kad parametrų tipai dalykinėms sritims yra bendri, skiriasi tik jų reikšmės ir naudojamų parametrų kiekis. Atlikus komunikacinių veiksmų kilpų parametrų parinkimą naudojantis gautais parametrų rinkiniais galima suprojektuoti DB struktūrą reikalingą kilpomis užduotoms transakcijoms vykdyti. Kadangi šiame darbe priimta prielaida, kad DB struktūra jau yra sudaryta, tuomet DB struktūros sudarymas iš nustatytų parametrų toliau neaprašomas.

3.3. **DB struktūrų atitikimo analizė užduotam paslaugos tipui**

Keičiantis verslo aplinkai gali keistis ir įmonių teikiamų paslaugų, parduodamų prekių tipai. Įmonės turi nusistovėjusias duomenų struktūras, tačiau keičiantis verslo profiliui esamos struktūros gali būti netinkamos naujų arba pakitusių verslo procesų vykdymui.

Norint nustatyti duomenų bazės struktūrų atitikimą paslaugos tipui reikia nustatyti naujai ar pakitusiai paslaugai reikalingą duomenų struktūrą ir palyginti ją su esama struktūra. Nustačius skirtumus tarp esamos ir reikiamos DB struktūrų galima priimti sprendimą ar keisti esamą struktūrą ar sukurti naują. Jeigu paslauga nežymiai pasikeitė tuomet yra paprasčiau papildyti (pakeisti) esamą struktūrą. Kitu atveju galima sukurti naują DB struktūrą. Kartais laiko sąnaudos gali būti mažesnės kuriant naują struktūrą nei keičiant esamą bei pritaikant ją verslo proceso vykdymui.

Nustatant pasikeitusios arba naujos paslaugos DB struktūrą galima naudotis 3.2 skyriuje aptartu būdu. Vietoje komunikacinių kilpų, kuriomis aprašomas verslo procesas galima naudoti detalų tekstinį paslaugos aprašymą (pakankamą verslo informacijai identifikuoti) ir pagal jį nustatinėti verslo informacijos komponentines grupes. Rezultate gaunamas parametrų sąrašas, kurį galima panaudoti nustatant struktūrų atitikimą. Tam, kad paslauga atitiktų DB struktūrą, esama struktūra turi padengti gautus parametrus, t.y. turi būti įmanoma išsaugoti parametrų reikšmes turint esamą DB struktūrą. Galimi tokie rezultatai:

- DB struktūra tiksliai atitinka paslaugos tipą. Paslaugai vykdyti nereikia keisti ar papildyti esamos DB struktūros.
- DB struktūroje trūksta keleto atributų parametrų reikšmėms saugoti. DB struktūra papildoma trūkstamais atributais. Pavyzdžiui, įmonė pradėjo pardavinėti naujas prekes ir užsakyme be visų esamų parametrų papildomai reikia nurodyti prekės spalvą. DB struktūra papildoma atributu *spalva*.
- DB struktūroje yra atributų, kurie nereikalingi paslaugos vykdymui. DB struktūroje gali būti panaikinami pertekliniai atributai arba struktūra paliekama be pakeitimų jei tie atributai reikalingi kitų (esamų) verslo procesų vykdymui. Pavyzdžiui, buvo pardavinėjamos prekės, kurių užsakyme reikėjo nurodyti matmenis. Atsiradus naujai prekei matmenų nurodyti nebereikia. Jeigu ir toliau pardavinėjama esama prekė, tuomet matmenų atributai paliekami, kitu atveju gali būti panaikinami iš DB struktūros.
- DB struktūroje trūksta didesnės dalies atributų. Gali būti kuriama nauja DB struktūra arba esama struktūra papildoma naujais struktūriniais elementais (atributais ir lentelėmis), kurie bus naudojami naujai paslaugai vykdyti. Sprendimo pasirinkimas priklauso nuo esamos DB struktūros ir naujos paslaugos skirtumo dydžio. Jeigu senos

paslaugos keičiamos naujomis žymiai besiskiriančiomis nuo esamų, tuomet iš esmės keičiasi DB struktūra ir būtina sudaryti naują struktūrą. Pavyzdžiui, jeigu įmonė prekiavusi stikliniais dirbiniais pradeda prekiauti tik stiklu. Tuomet DB struktūros žymiai skirtųsi ir tektų sudaryti naują DB struktūrą.

Atlikus DB struktūrų pritaikymą naujai paslaugai gali tekti keisti arba papildyti ir sistemos valdančios transakcijas funkcijas.

3.4. **Duomenų bazės transakcijų sudarymas**

Duomenų bazės transakcijų sudarymas - tai perėjimas nuo verslo transakcijų prie DB transakcijų. Vizualiai pateiktas verslo proceso vykdymas paverčiamas konkrečiomis DB užklausomis. Parametrizuotos komunikacinės veiksmų kilpos yra pagrindas duomenų bazės transakcijų sudarymui. Nustatyti kilpų parametrai tiesiogiai panaudojami DB transakcijose kaip užklausų parametrai.

Naudojantis komunikacinių kilpų parametrais sudaromos DB užklausos. Užklausų sudarymui naudojamas bendras automatizuotas mechanizmas. Būtina pabrėžti, kad verslo transakcijai atlikti reikalingos DB užklausos yra aprašomos sudarant programą, kuri vykdo verslo transakcijas, t.y. DB užklausos deklaruojamos iš anksto ir jau nekintamos programos vykdymo metu. DB užklausos nėra rašomos tiesiogiai, o pateikiamos kreipiniais į užklausų sudarymo mechanizmą, kuriam paduodami parinkti parametrų rinkiniai ar jų dalis. Tokiu būdu esant programoje aprašytiems kreipiniams į užklausų sudarymo mechanizmą, užklausų generavimas atliekamas realiu laiku vykdant atitinkamus verslo transakcijos veiksmus.

Užklausų sudarymo mechanizmas yra naudojamas tam, kad pasikeitus ar atsiradus naujiems paslaugų tipams ir pasikeitus duomenų bazės struktūrai nereikia perrašyti ar rašyti naujų DB užklausų. Esant pasikeitimams keičiasi tik parametrų rinkiniai, kurie bendro mechanizmo yra vienodai interpretuojami, kaip ir esamų paslaugų parametrai, ir panaudojami sudarant užklausas.

Sudarant užklausas išskiriami trys DB modifikavimo užklausų tipai: *insert*, *update* ir *delete*. Šių užklausų atlikimui gali būti reikalinga ir duomenų išrinkimo užklausa *select*, kurios rezultatas gali būti panaudojamas modifikavimo užklausose kaip vienas iš parametrų. Kadangi užklausų sudarymo mechanizmas yra bendras įvairių paslaugų tipams, tuomet jis turi būti bendras tiek skirtingoms duomenų bazių struktūroms, tiek skirtingų parametrų rinkiniams, tiek visų užklausų tipams.

Į užklausų sudarymo mechanizmą paduodami įėjimo parametrai ir gaunamas rezultatas – sudaryta užklausa pagal gautus parametrus. Užklausų sudarymo mechanizmo funkcijos antraštė (PHP kalboje):

function queryGenerator (*$queryType*, *$tableName*, *$queryParameters*, *$restrictions*)

Pirmasis parametras (*$queryType*) nusako pačios užklausos tipą. Skirtingų tipų užklausoms sudaryti reikalingi skirtingi papildomi įėjimo parametrai:

- *Insert* užklausos parametrai:
  - *$tableName* – lentelės, į kurią įterpiami duomenys, pavadinimas
  - *$queryParameters* – tai verslo transakcijos parametrai ar jų dalis kartu su tų parametrų reikšmėmis, kurios įterpiamos į nurodytą lentelę
- *Update* užklausos parametrai:
  - *$tableName* – lentelės, kurios duomenys atnaujinami, pavadinimas
  - *$queryParameters* – tai verslo transakcijos parametrai ar jų dalis kartu su tų parametrų reikšmėmis, kurios atnaujinamos nurodytoje lentelėje
  - *$restrictions* – nurodo atnaujinamų įrašų aibę (užklausoje atitinka *WHERE*, *ORDER BY* ir *LIMIT* dalį)
- *Delete* užklausos parametrai:
  - *$tableName* – lentelės, iš kurios šalinami duomenys, pavadinimas
  - *$restrictions* – nurodo šalinamų įrašų aibę (užklausoje atitinka *WHERE*, *ORDER BY* ir *LIMIT* dalį)
- *Select* užklausos parametrai:
  - *$tableName* – lentelės(ių), iš kurios(ių) išrenkami duomenys, pavadinimas(ai)
  - *$queryParameters* – tai atributai, kurių reikšmės išrenkamos iš nurodytos(ų) lentelės(ių)
  - *$restrictions* – nurodo išrenkamų įrašų aibę (užklausoje atitinka *WHERE* dalį)

Jeigu užklausos sudarymui nenaudojamas kuris nors parametras, tuomet jo vietoje įrašoma tuščia reikšmė.

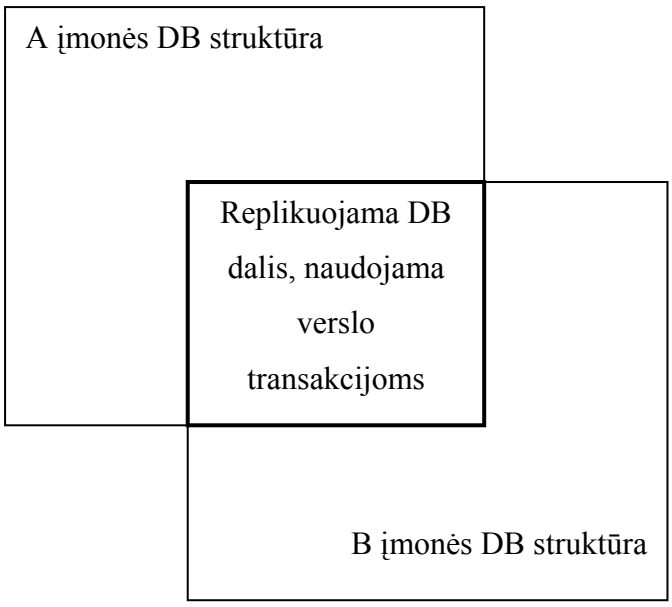
Gauta sudaryta užklausa yra sintaksiškai tvarkinga ir gali būti vykdoma. Užklausos tvarkingą sudarymą garantuoja automatizuotas užklausų sudarymo mechanizmas, kuris tikrina gaunamus parametrus ir trūkstant arba esant neteisingai nurodytiems parametrams grąžinamas klaidos pranešimas.

Verslo transakcijai įvykdyti reikia bent dviejų (viename serveryje) arba keturių (dviejuose serveriuose) DB transakcijų. Tuo tarpu vykstant kiekvienam transakcijos veiksmui gali prireikti atlikti keletą DB transakcijų. Taigi verslo transakciją gali sudaryti neribotas kiekis DB transakcijų,

kurių kiekviena sukuriama naudojant užklausų sudarymo mechanizmą. Gautos užklausos yra vykdomos viena po kitos.

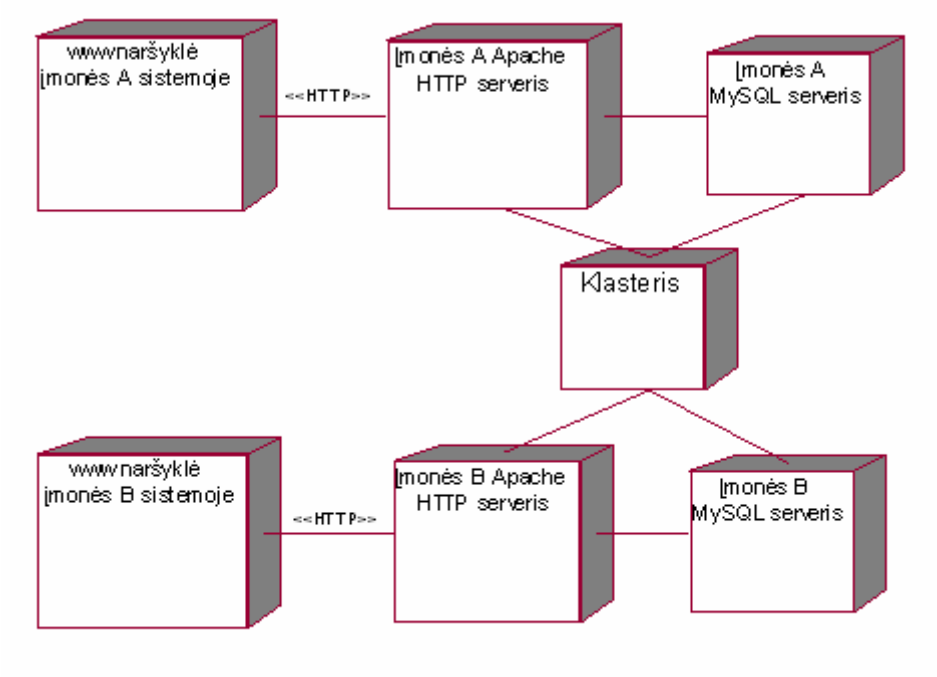
3.5. **Transakcijų realizavimas replikuojančiose DB**

Dauguma įmonių turi dideles individualias duomenų bazes, kurios naudojamos įmonės reikmėms. Dalis duomenų bazės gali būti naudojama įmonės vidinei veiklai palaikyti, o kita dalis gali būti naudojama verslo transakcijų palaikymui. Jeigu verslo reikalais bendradarbiauja dvi tokios įmonės, kurios turi dideles duomenų bazes, tuomet atsiranda DB replikavimo poreikis. Naudojamas replikavimas jungties pagrindu siekiant padidinti replikavimo sistemos lankstumą bei saugoti mažiau duomenų. Bendradarbiaujant tokioms įmonėms dalis jų DB persidengia (*17 paveikslas*), t.y. dalyje lentelių saugoma ta pati informacija.



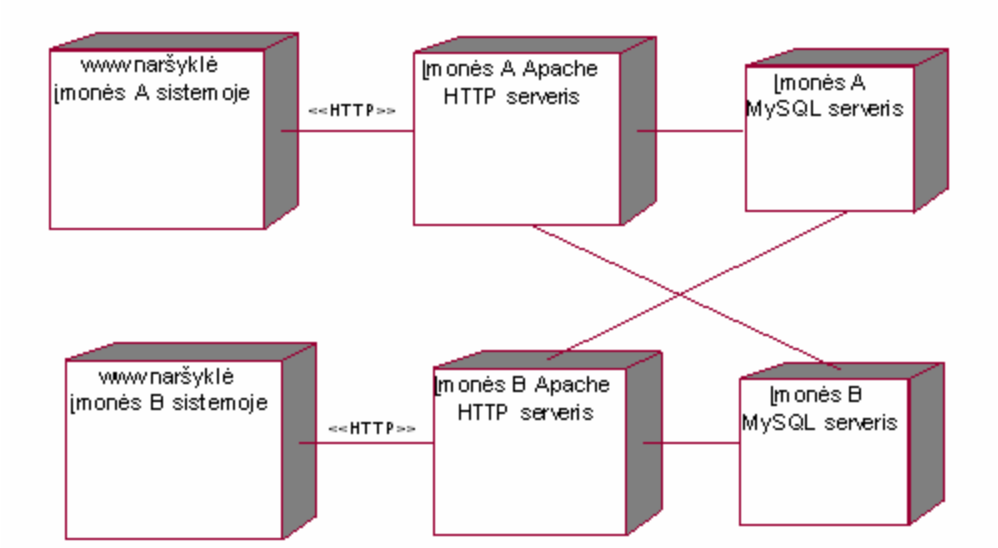
17 pav. DB replikavimas

Bendru atveju gali skirtis persidengiančios duomenų struktūros, tačiau informacija saugoma ta pati. Tokiu atveju naudojamas klasteris(iai) (DB lentelių, atributų sinonimų sąrašas), kuris valdo informacijos srautus ir leidžia skirtingose duomenų struktūrose saugoti tą pačią informaciją. Jeigu abiejose įmonėse naudojami Apache HTTP serveriai ir MySQL duomenų bazės, tuomet įdiegimo diagrama atrodytų taip, kaip pavaizduota *18 paveiksle*.



18 pav. Replikuojančios sistemos įdiegimo diagrama panaudojant klasterį

Nagrinėsime atvejį kuomet replikuojančių DB struktūros yra vienodos ir klasteris nėra naudojamas (*19 paveikslas*).



19 pav. Replikuojančios sistemos įdiegimo diagrama nenaudojant klasterio

Pagrindinis uždavinys yra užtikrinti, kad duomenų bazių persidengiančiose lentelėse būtų saugoma ta pati informacija. Jeigu duomenų bazės yra skirtinguose serveriuose, tuomet viena jų arba abi kartu gali būti laikinai nepasiekiamos. Tokiu atveju būtina užtikrinti, kad DB transakcijos bus vienodai įvykdytos tiek vienoje, tiek kitoje DB. Nutrūkus ryšiui arba nepavykus prisijungti bent prie vieno serverio duomenų bazės, DB transakcijos apskritai neturi būti vykdomos.

Tarkime, kad turime dvi skirtinguose serveriuose esančias duomenų bazes DB1 ir DB2. Jeigu lokaliame serveryje su duomenų baze DB1 pradedamas verslo transakcijos veiksmas atnaujinantis duomenis abiejose duomenų bazėse, tuomet sinchronizaciją galima užtikrinti toliau aprašytu būdu:

- atnaujinant duomenis reikalingos dvi SQL užklausos. Pirmoji užklausa atnaujina įrašus lokalioje DB1 duomenų bazėje, o antroji – išorinėje DB2 duomenų bazėje. Antroji užklausa vykdoma tik tuo atveju jeigu sėkmingai įvyksta pirmoji.

- pirmiausiai bandoma prisijungti prie lokalios DB1 duomenų bazės ir joje įvykdyti DB užklausą. Jeigu užklausa neįvyksta (nutrūko ryšys arba sustabdytas serveris), tuomet transakcija nutrūksta pačioje pradžioje.

- jeigu pavyksta įvykdyti užklausą lokalioje duomenų bazėje DB1, tuomet bandoma vykdyti užklausą išorinėje DB2 duomenų bazėje.

- jeigu nepavyksta įvykdyti užklausos išorinėje duomenų bazėje DB2 (nes serveris sustabdytas arba nutrūko ryšys su išoriniu serveriu), atstatomos DB1 modifikacijos, kurias atliko lokalioje duomenų bazėje įvykdyta užklausa.

- jeigu užklausa išorinėje duomenų bazėje DB2 sėkmingai įvykdoma, tuomet DB transakcijos vykdymas sėkmingai užbaigiamas.

Tokiu būdu garantuojama DB transakcijos sinchronizacija. Užtikrinama, kad DB transakcija įvyks abiejose duomenų bazėse arba nebus įvykdyta nei vienoje duomenų bazėje.

Jeigu verslo transakcijos veiksmui atlikti reikalingos kelios DB transakcijos, tuomet užklausų vykdymas atliekamas analogiškai, kaip ir esant vienai DB transakcijai. Iš pradžių atliekamos visos užklausos lokalioje duomenų bazėje, o po to išorinėje duomenų bazėje.

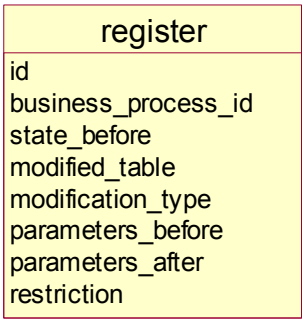
Aprašytai transakcijų sinchronizacijai užtikrinti naudojamas DB transakcijų sinchronizavimo mechanizmas, kuris turi vieną įėjimo parametrą – reikiamų atlikti užklausų sąrašą. Užklausų korektiškumą užtikrina 3.4 skyriuje aprašytas užklausų sudarymo mechanizmas.

3.6. **Įvykdytų duomenų bazės transakcijų atstatymas**

Vykstant verslo procesui jis bet kuriuo metu gali būti nutrauktas. Iki nutraukimo momento gali būti jau įvykdyta keletas verslo procesą sudarančių transakcijų. Nutraukus verslo procesą reikia atstatyti įvykdytas verslo transakcijas. Kitaip sakant, kadangi verslo transakcijų metu buvo įvykdytos jas atitinkančios duomenų bazės transakcijos, tuomet reikia atstatyti duomenų bazės užklausų atliktus pakeitimus duomenų bazėje(se).

Tam, kad būtų galima vykdyti duomenų bazės transakcijų atstatymą, prieš tai jos turi būti registruojamos. Atliktų DB užklausų registravimui naudojamas užklausų žurnalas. Užklausų registravimas glaudžiai susijęs su užklausų sudarymo mechanizmu. Žurnalo pildymui panaudojami tie patys parametrų rinkiniai, kurie naudojami užklausų sudarymui, todėl užklausos registravimas žurnale atliekamas iškart po užklausos sudarymo ir jos įvykdymo.

Užklausų saugojimui naudojama viena DB lentelė *Register*, kurios struktūra pateikta *20 paveiksle*.



20 pav. Žurnalo lentelės struktūra

Registro lentelėje atliktoms duomenų bazės transakcijoms saugoti naudojami atributai:

- *id* – registro įrašo identifikatorius,
- *business_process_id* – verslo proceso, su kurio transakcija susijusi duomenų bazės transakcija, identifikatorius,
- *state_before* – būseną, kurioje prieš atliekant modifikuojančią užklausą buvo verslo procesas, identifikuojančių parametrų rinkinio dalis (atributas paliekamas tuščias, jeigu verslo proceso pradžioje nebuvo prieš būsenos). Šis atributas saugo *exist* būsenų priklausomybės parametrus,
- *modified_table* – lentelės, kurioje atliekama modifikuojanti užklausa, pavadinimas,
- *modification_type* – atliekamos modifikuojančios užklausos tipas (*insert*, *update*, *delete*),
- *parameters_before* – verslo transakcijos parametrai ar jų dalis kartu su tų parametrų reikšmėmis, kurios buvo prieš atliekant užklausą (ne visada būtinas, žiūrėti *6 lentelę*),
- *parameters_after* – verslo transakcijos parametrai ar jų dalis kartu su tų parametrų reikšmėmis, kurios buvo įterptos/atnaujintos po užklausos įvykdymo (ne visada būtinas, žiūrėti *6 lentelę*),
- *restriction* – modifikuojančios užklausos apribojimai, kurie užklausoje atitinka *WHERE*, *ORDER BY*, *LIMIT* dalis.

Registro lentelės atributai *parameters_before* ir *parameters_after* naudojami ne visų tipų užklausų saugojimo metu. Šių lentelės atributų panaudojimas aprašytas *6 lentelėje*.

6 lentelė. Parametrų panaudojimas registruojant užklausas

Modifikuojančios užklausos tipas	parameters_before naudojimas	parameters_after naudojimas	Paaiškinimas
Insert	-	+	Įterpiamas įrašas panaudojant gautus parametrus, kurie išsaugomi atribute *parameters_after*. Esami duomenų bazės duomenys nemodifikuojami, todėl *parameters_before* atributas nenaudojamas (reikšmė *NULL*)
Update	+	+	Atnaujinami įrašai panaudojant gautus parametrus, kurie išsaugomi atribute *parameters_after*. Duomenys (parametrų reikšmių rinkiniai) buvę prieš atnaujinimą išsaugomi atribute *parameters_before*
Delete	+	-	Ištrinamų įrašų duomenys (parametrų reikšmių rinkiniai) išsaugojami *parameters_before* atribute. Ištrynus įrašus kitaip duomenų bazė nemodifikuojama, todėl *parameters_after* atributas nenaudojamas (reikšmė *NULL*)

Užklausų žurnalas pildomas kol verslo procesas galutinai užbaigiamas arba nutraukiamas. Jeigu verslo procesas sėkmingai užbaigiamas, tuomet užklausų žurnalas išvalomas, nes baigtas procesas jau nebus atšaukiamas ir jau nereikalinga atliktų užklausų istorija. Registro išvalymas įvykdomas viena *Delete* užklausa ištrinant visus įrašus, kurių *business_process_id* lygus užbaigto verslo proceso identifikatoriui. Jeigu verslo procesas nutraukiamas, tuomet užpildytas užklausų žurnalas panaudojamas duomenų bazės atstatymui.

Atstatant duomenų bazę pirmiausia iš užklausų žurnalo išrenkamas verslo proceso vykdymo metu įvykdytų užklausų sąrašas. Užklausų istorija yra vienintelis atliktų duomenų bazės modifikacijų šaltinis. Naudojantis užklausų istorija, kiekvienai atliktai DB transakcijai (užklausai) sudaroma kompensacinė transakcija. Pagal užklausų žurnalo lentelės atributą *modification_type*

nustatomas įvykdytos užklausos tipas. Skirtingiems užklausų tipams sudaromos skirtingų tipų kompensuojančios užklausos panaudojant skirtingų žurnalo lentelės atributų reikšmes:

- *Insert užklausos kompensacija*:

  Sudaroma *Delete* tipo kompensacinė užklausa. Ištrinamas įrašas iš lentelės *modified_table*. Užtikrinant, kad būtų ištrintas reikiamas įrašas, panaudojamas parametrų rinkinys *parameters_after* bei apribojimų atributo *restriction* reikšmė. Parametrų reikšmės (jų dalis) ir apribojimai sudarant kompensacinę užklausą įterpiami į užklausos WHERE dalį ir tokiu būdu nufiltruojamas reikiamas įrašas. Paprasčiausiu atveju filtrui užtenka įterpto įrašo identifikatoriaus.

- *Update užklausos kompensacija*:

  Sudaromos *Update* tipo kompensacinės užklausos. Atnaujinami įrašai lentelėje *modified_table*. Užtikrinant, kad būtų atnaujinti reikiami įrašai, panaudojamas parametrų rinkinys *parameters_after* bei apribojimai (atributas *restriction*). Parametrų reikšmės (jų dalis) bei apribojimai sudarant kompensacinę užklausą įterpiami į užklausos WHERE dalį ir tokiu būdu nufiltruojami bei atnaujinami reikiami įrašai. Paprasčiausiu atveju filtrui užtenka atnaujintų įrašų identifikatorių. Atnaujinama atitinkamai pagal atributo *parameters_before* saugomus senus kiekvieno įrašo duomenis.

- *Delete užklausos kompensacija*:

  Sudaromos *Insert* tipo kompensacinės užklausos. Įrašai įterpiami į lentelę *modified_table*. Ištrintų įrašų atstatymui panaudojamas parametrų rinkinys *parameters_before*. Įrašai įterpiami atitinkamai pagal atributo *parameters_before* saugomus senus kiekvieno įrašo duomenis. Filtrai įterpiant įrašus nereikalingi.

Sudarytos užklausos atstato duomenų bazės duomenis. Visais kompensuojančių užklausų sudarymo atvejais sudaroma ir papildoma būsenos atstatymo kompensuojanti užklausa, kurios kūrimui panaudojami atributo *state_before* saugomi parametrai išlaikantys *exist* priklausomybes. Būsenos atstatymas leidžia patikimai vykdyti sudėtinių transakcijų susietų *exist* priklausomybėmis atstatymą. Sudarius kompensacines užklausas pradedamas jų vykdymas. Kompensacinės užklausos vykdomos tokiu pat būdu kaip ir duomenų bazių transakcijos atitinkančios verslo transakcijas. Esant replikuojamoms duomenų bazėms 3.5 skyriuje aptartu būdu užtikrinama DB atstatymo sinchronizacija. Vykdant kompensacines užklausas tuo pačiu išvalomas ir užklausų žurnalas.

Bendru atveju duomenų bazės atstatymą po nutraukto verslo proceso T galima vykdyti naudojantis tokiu algoritmu:

Atšaukti[T]:

1. Nutraukti visus veiksmus susijusius su verslo procesu T.
2. Nustatyti verslo proceso T būseną į "atšaukiamas".
3. Gauti (sudaryti) visas neįvykdytas verslo proceso kompensacines užklausas. Vykdyti žemiau esančius žingsnius a-c atliekant kompensacines užklausas.
    a. Pradėti kompensacinės užklausos vykdymą.
    b. Pašalinti DB transakcijos įrašą iš užklausų žurnalo.
    c. Jeigu *a* arba *b* žingsnyje nepavyksta įvykdyti kompensacinės transakcijos (nes serveris sustabdytas arba nutrūko ryšys su vienu iš serverių), tuomet verslo proceso būsena nustatoma į "nepavykęs atstatymas". Likusios kompensacinės transakcijos nevykdomos. Proceso atstatymą galima pratęsti vėliau (kai pasiekiami abu serveriai) naudojantis tais pačiais žingsniais.
4. Patvirtinti verslo proceso atšaukimą.

Įvykdžius visas kompensacines DB transakcijas verslo procesas tampa visiškai pašalintas iš duomenų bazės. Tik sėkmingai atlikus duomenų bazės atstatymą išlaikomas DB pilnumas ir nuoseklumas.

4. VERSLO TRANSAKCIJŲ VYKDYMO ELEKTRONINIŲ PASLAUGŲ SRITYJE EKSPERIMENTINIS TYRIMAS

4.1. Dalykinės srities apibrėžimas

Sparčiai vystantis internetinėms technologijoms, didėja elektroninių paslaugų spektras. Darbo eksperimentiniam tyrimui atlikti pasirinkta konkreti elektroninių paslaugų sritis – internetinių svetainių kūrimas. Internetinės svetainės kūrimo procesą sudaro užsakymo priėmimas, sutarties sudarymas, projekto derinimas, vykdymas bei apmokėjimas. Procesas yra visiškai kompiuterizuotas ir vykdomas internetinėje erdvėje.

Internetinės svetainės kūrimo procese dalyvauja dvi šalys:

- klientas (fizinis arba juridinis asmuo), kuris užsisakė paslaugą. Kliento tikslas yra sukurtas internetinis puslapis pagal jo pateiktus reikalavimus.
- įmonė, kuri atlieka paslaugos vykdymą. Įmonės tikslas yra apmokėjimo gavimas už atliktą paslaugą (internetinės svetainės sukūrimą).

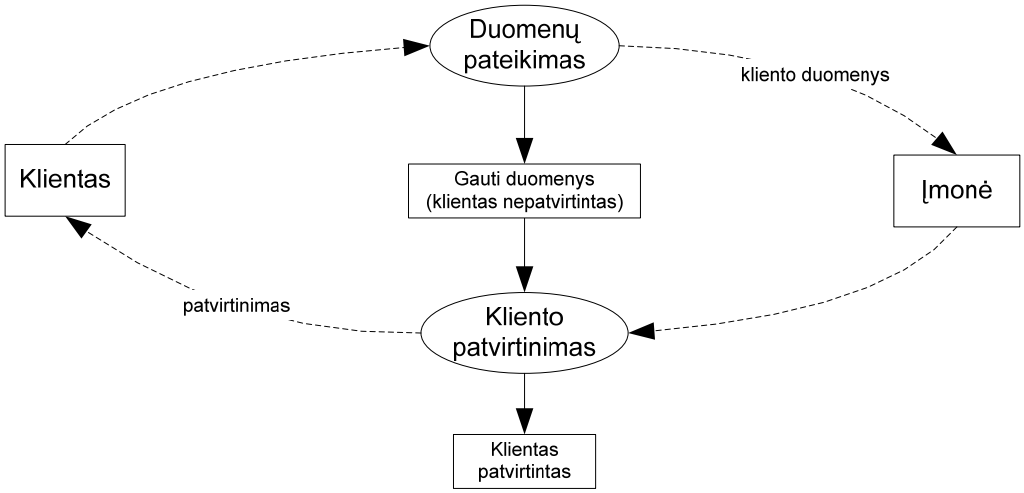
Sėkmingai įvykdytas paslaugos užsakymo procesas garantuoja abiejų šalių tikslų įgyvendinimą. Nutraukus užsakymo vykdymą, jis pašalinamas iš įmonės užsakymų sąrašo.

Eksperimente naudojamos dvi replikuojančios duomenų bazės. Viena jų yra bendrame klientų serveryje, prie kurio jungiasi visi klientai, o kita – lokaliame įmonės serveryje, prie kurio jungiasi tik toji įmonė.

Eksperimente didžiausias dėmesys skiriamas komunikacinėmis kilpomis užduotų verslo transakcijų realizavimui bei jų atšaukimui verslo proceso nutraukimo atveju.

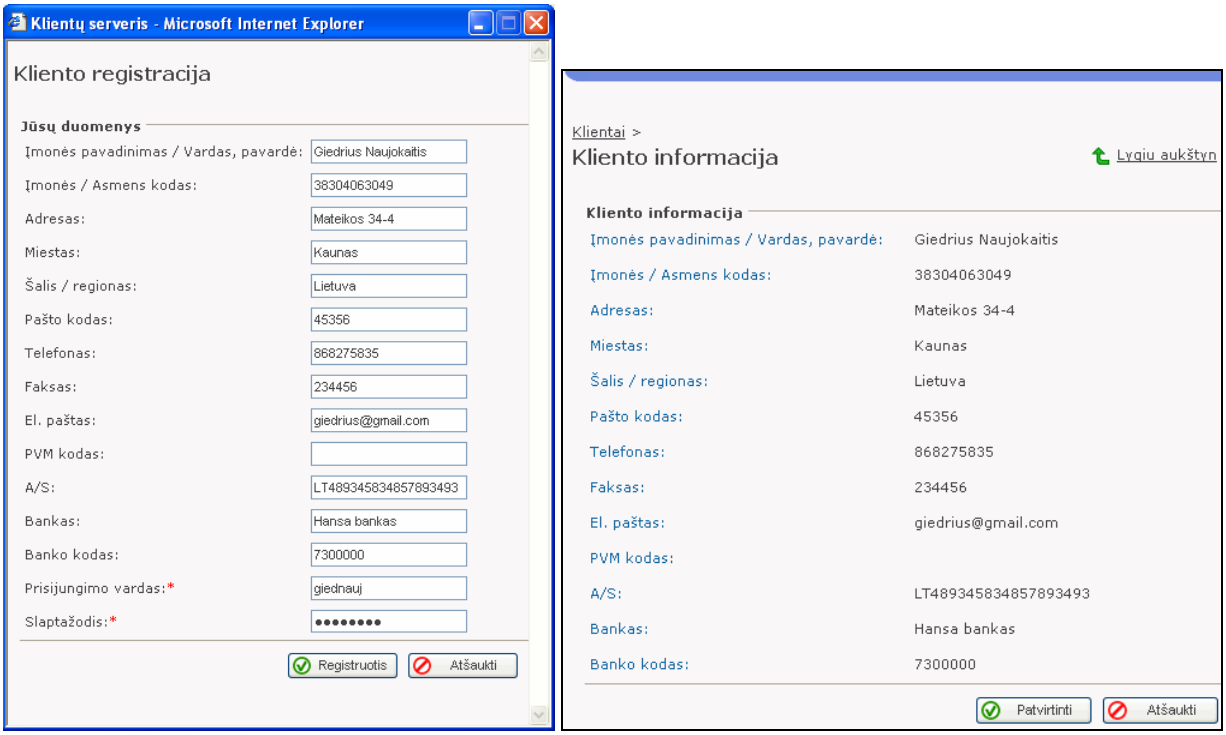
4.2. Eksperimentinis verslo transakcijų vykdymas

Klientas, norėdamas užsisakyti internetinės svetainės kūrimo paslaugą, pirmiausiai turi užsiregistruoti sistemoje. Klientas registruojasi bendrame klientų serveryje. Kliento registracijos komunikacinė kilpa pateikta *21 paveiksle*.



21 pav. Kliento registracijos komunikacinė kilpa

Klientui užsiregistravus jo informacija tampa matoma įmonei. Tuomet įmonės atsakingas atstovas nusprendžia (jeigu reikia, tikrina informaciją remdamasis trečiomis šalimis), ar patvirtinti naujo kliento registraciją, ar ne. Registracijos ir jos patvirtinimo langai pateikti *22 paveiksle*.

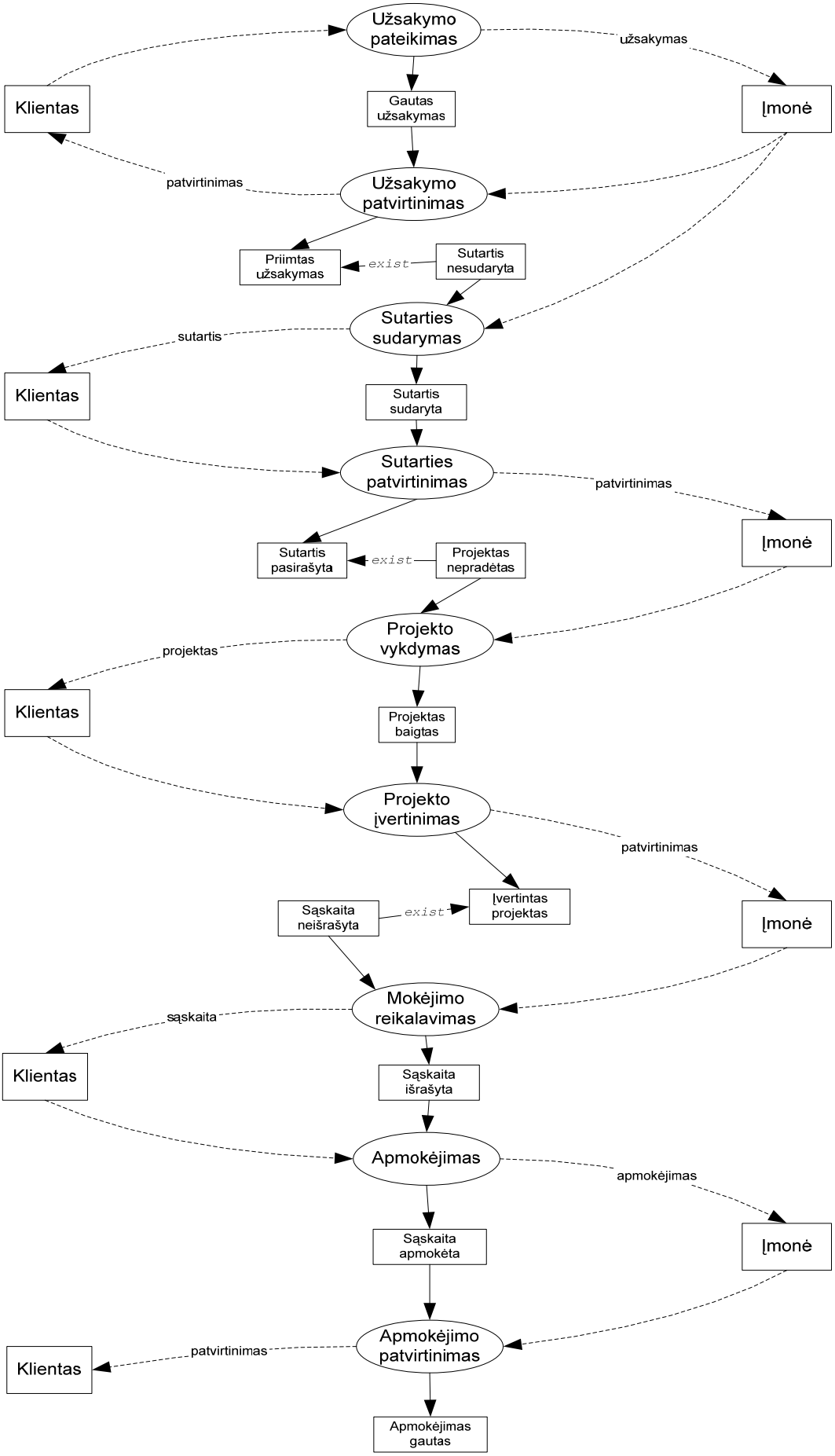


a) ***b)***

22 pav. a) Kliento registracijos langas; b) Kliento registracijos patvirtinimo langas

Priėmus arba atmetus kliento registraciją, klientui siunčiamas pranešimas apie įmonės nuosprendį. Jeigu kliento registracija patvirtinama, tuomet jis gali prisijungti prie sistemos ir užsisakyti norimą paslaugą. Kitu atveju kliento registracijos informacija pašalinama.

Internetinės svetainės kūrimo proceso sudėtinė komunikacinė kilpa pateikta *23 paveiksle*.



23pav. Internetinės svetainės kūrimo proceso komunikacinė kilpa

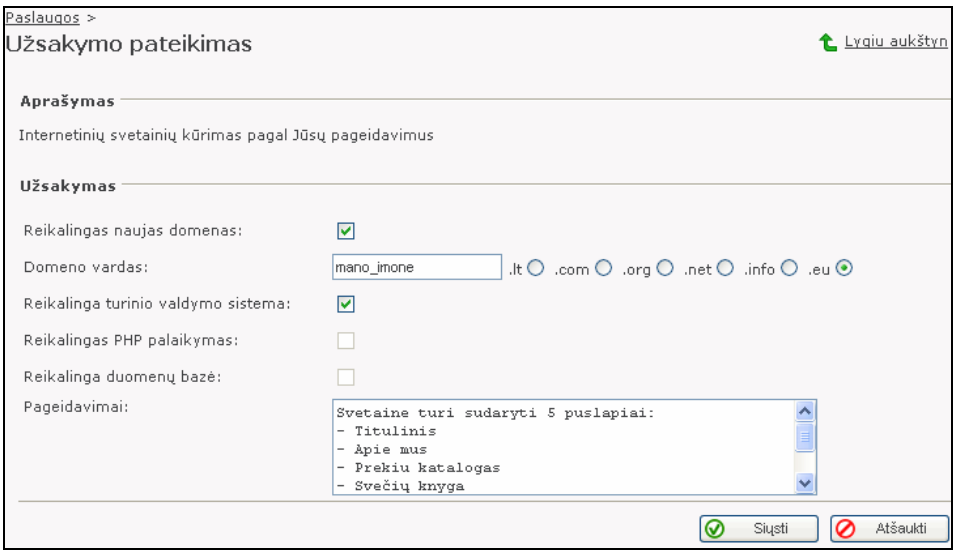
Internetinės svetainės kūrimo procesas yra sudėtinis, t.y. sudarytas iš keleto verslo transakcijų, atliekamų viena po kitos. Vykstant svetainės kūrimo procesui bet kuriuo metu jis gali būti nutrauktas paspaudus kiekviename toliau pavaizduotame lange esantį mygtuką „Atšaukti“. Vykstant verslo transakcijoms vis didėja su proceso nutraukimu susijusi atsakomybė. Pradžioje, pateikus užsakymą ir jo atsisakius iki projekto vykdymo pradėjimo, nenešama jokia atsakomybė, tačiau vėliau (pavyzdžiui, nutraukiant procesą kuomet pasirašyta sutartis ir projektas įvykdytas, o apmokėjimas negautas) klientas gali būti patrauktas teisinėn atsakomybėn įsiterpiant trečiosioms šalims. Todėl proceso nutraukimas vykdomas tik esant svarioms priežastims.

Internetinės svetainės kūrimo procesas vykdomas atliekant jį sudarančių transakcijų veiksmus (verslo transakciją sudaro du veiksmai: inicijuojamas ir grįžtamojo ryšio veiksmas). Kad būtų galima vykdyti atitinkamą veiksmą, procesas turi būti tam tikroje būsenoje, kuri yra atitinkamo veiksmo prieš-būsena išskyrus pirmąjį veiksmą (*užsakymo pateikimas*), kurio metu yra sukuriamas objektas (užsakymas).

Internetinės svetainės kūrimo proceso veiksmai:

- *Užsakymo pateikimas* – klientas pateikdamas užsakymą turi užpildyti *24 paveiksle* pavaizduotą formą, į kurią surašomi kliento pageidavimai norimam projektui:
    - *Reikalingas naujas domenas* – pažymimas, jeigu klientas neturi savo domeno interneto svetainei saugoti. Jeigu reikalingas domenas, tuomet užpildomas *domeno vardo* laukelis;
    - *Domeno vardas* - įvedamas domeno vardas ir parenkamas jo tipas. Nepildomas, jeigu nereikalingas naujas domenas;
    - *Reikalinga turinio valdymo sistema* – pažymimas jei klientas pageidauja turėti svetainės turinio valdymo sistemą (*TVS*). Pasirinkus TVS, PHP ir duomenų bazės laukeliai nepildomi;
    - *Reikalingas PHP palaikymas* - nurodo ar reikalingas PHP palaikymas. Neparinkus bus kuriamas tik HTML puslapis;
    - *Reikalinga Duomenų bazė* - nurodo ar klientas nori turėti duomenų bazę.
    - *Pageidavimai* – surašomi pageidaujamos svetainės reikalavimai ir kt.

  Klientas pateikia savo pageidavimus ir siunčia juos įmonei.



24 pav. Užsakymo sudarymo langas

Prieš būsenos parametrai:

Užsakymo pateikimo veiksmas neturi prieš būsenos, nes atliekant veiksmą sukuriamas naujas objektas (užsakymas).

Srauto parametrai:

Užsakymo pateikimo srautą sudaro tokie parametrai:

- *order_number* – automatiškai sugeneruojamas unikalus pateikiamo užsakymo numeris,
- *client_id* – kliento, kuris pateikė užsakymą identifikatorius,
- *client_name* – pateikusio užsakymą kliento pavadinimas/vardas su pavarde,
- *service_id* – užsisakytos paslaugos identifikatorius,
- *date_created* - užsakymo pateikimo data,
- *domain_required* – nurodo ar reikalingas naujas domenas,
- domain_name – naujo domeno vardas (jeigu reikalingas domenas),
- domain_type – naujo domeno tipas (jeigu reikalingas domenas),
- cms_required – nurodo ar reikalinga turinio valdymo sistema,
- php_support – nurodo ar reikalingas PHP palaikymas,
- db_required– nurodo ar reikalingas duomenų bazė,
- request – pageidavimai paslaugos atlikimui.

Naudojantis šiais parametrais sudaromos dvi *Insert* užklausos. Pirmajai panaudojami pirmieji penki parametrai, kurie įterpiami į užsakymų lentelę (*orders*), o likusieji septyni įterpiami į užsakymo informacijos lentelę (*order_details*).

Visais atvejais *Insert* užklausos struktūra yra vienoda:

INSERT INTO table (field_1, field_2,...) VALUES (value_1, value_2,...)

Parametrų pavadinimai atitinka duomenų bazės laukus (*field_#*), o jų reikšmės atitinka į duomenų bazę įrašomas reikšmes (*value_#*).

Taigi sudaryta pirmoji užklausa bus tokia:

INSERT INTO orders (order_number, client_id, client_name, service_id, date_created) VALUES (' 6vmdxpht5v73ea9fvp9t4v3x', 11, 'Giedrius Naujokaitis', 1, '2006-11-27')

Analogiškai sudaromos visose verslo transakcijose vykdomos *Insert* užklausos ir tolesniame aprašyme jos nedetalizuojamos.

Po būsenos parametrai:

Pateikus užsakymą pradėtas verslo procesas pereina į būseną „*Gautas užsakymas*“. Būsenos parametrai:

- *order_id* – pateikto užsakymo identifikatorius,
- *order_number* – pateikto užsakymo numeris,
- *client_id* – pateikusio užsakymą kliento identifikatorius,
- *order_confirmed* – nurodo pateikto užsakymo būseną (ar patvirtintas).

Pateikus užsakymą verslo procesas bus būsenoje, kurią galima aprašyti vektoriumi *B*:

B{42, '6vmdxpht5v73ea9fvp9t4v3x', 11, false}

Analogiškai aprašomos visos verslo proceso būsenos.

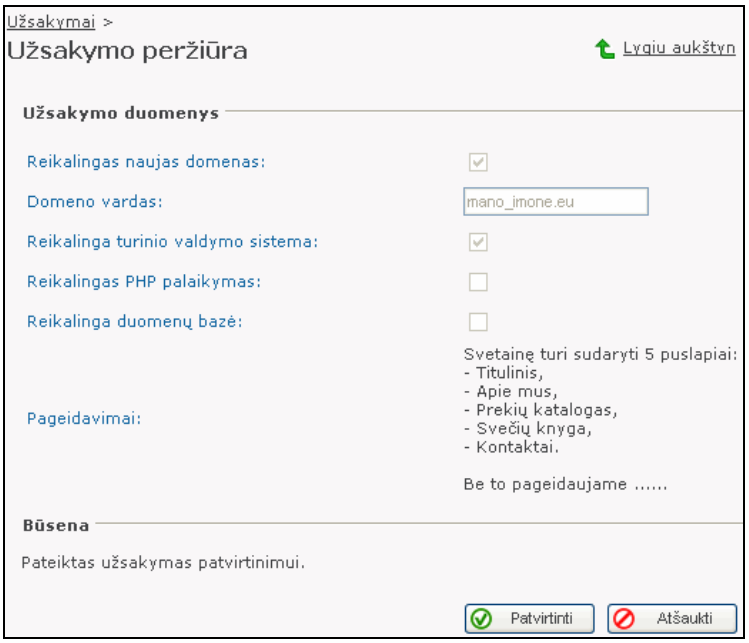
Gauti srautų bei būsenų parametrai surašomi į užklausų registrą, kuris prireikus bus panaudojamas duomenų bazės atstatymui. Registro laukų (*20 paveikslas*) užpildymas:

- *business_process_id* – įrašomas užsakymo identifikatorius: *42*,
- *state_before* – paliekamas tuščias, nes nėra prieš būsenos. Bendru atveju šiame lauke įrašomas prieš būsenos vektorius,
- *modified_table* – įrašomas lentelės pavadinimas: *orders*,
- *modification_type* – įrašomas užklausos tipas: *Insert*,
- *parametre_before* – paliekamas tuščias. Bendru atveju šiame lauke įrašomi parametrai *6 lentelėje* aprašytu būdu,
- *parameters_after* – įrašomas srauto parametrų rinkinys,

- *restriction* – įrašomas užsakymo numerio parametras: *order_number = '6vmdxpht5v73ea9fvp9t4v3x'*.

Užklausų registras panašiai pildomas visų transakcijų vykdymo metu.

- *Užsakymo patvirtinimas* – jeigu įmonė turi gautų užsakymų (prieš būsena), tuomet ji gali peržiūrėti užsakymus ir juos arba patvirtinti, arba atmesti. Užsakymo patvirtinimo langas pateiktas *25 paveiksle*



25 pav. Užsakymo patvirtinimo langas

Prieš būsenos parametrai:

Užsakymo patvirtinimo veiksmo prieš būsena sutampa su užsakymo pateikimo po būsena.

Srauto parametrai:

Užsakymo patvirtinimo srautą sudaro vienas parametras:

- *order_confirmed* – nurodo užsakymo patvirtinimą.

Naudojantis šiuo parametru sudaroma *Update* užklausa, kuri atnaujina užsakymo būseną užsakymų lentelėje (*orders*).

Visais atvejais *Update* užklausos struktūra yra vienoda:

UPDATE table SET field_1= value_1, field_2= value_2, ... WHERE restriction

Parametrų pavadinimai atitinka atnaujinamus duomenų bazės laukus (*field_#*), o jų reikšmės atitinka į duomenų bazę įrašomas reikšmes (*value_#*). Apribojimus (r*estriction*) atitinka būsenos parametrai ar jų dalis.

Taigi sudaryta užklausa bus tokia:

UPDATE orders SET order_confirmed = 'true' WHERE order_number = '6vmdxpht5v73ea9fvp9t4v3x'

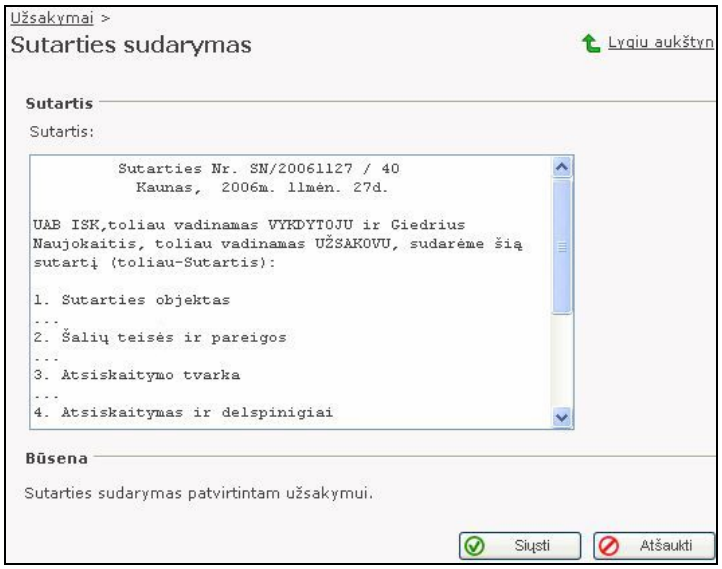
Analogiškai sudaromos visose verslo transakcijose vykdomos *Update* užklausos ir tolesniame aprašyme jos nedetalizuojamos.

Po būsenos parametrai:

Patvirtinus užsakymą verslo procesas pereina į būseną „*Priimtas užsakymas*“. Būsenos parametrai išlieka tokie patys kaip ir prieš būsenoje, tik pasikeičia parametro *order_confirmed* reikšmė:

B{42, '6vmdxpht5v73ea9fvp9t4v3x', 11, true}

- *Sutarties sudarymas* – patvirtinus užsakymą, jis tampa priimtu. Sutarties sudarymo etape sudaroma užsakymo vykdymo sutartis ir siunčiama klientui. Sutarties sudarymo langas pateiktas *26 paveiksle*.



26 pav. Sutarties sudarymo langas

Prieš būsenos parametrai:

Prieš sutarties sudarymą verslo procesas yra būsenoje „*Sutartis nesudaryta*“. Būsenos parametrai:

- *order_number* – užsakymo numeris,

- *contract_confirmed* – nurodo sudarytos sutarties būseną (ar pasirašyta).

Būsena „*Sutartis nesudaryta*“ susieta *exist* priklausomybe su būsena „*Priimtas užsakymas*“. Tai reiškia, kad turi egzistuoti būseną „*Priimtas užsakymas*“ aprašantis vektorius *B{42, '6vmdxpht5v73ea9fvp9t4v3x', 11, true}*, su kuriuo susijusi informacija jau yra įrašyta duomenų bazėje. *Exist* priklausomybės išlaikymui naudojamas parametras *order_number*.

Taigi sutarties sudarymo prieš būsenos vektorius yra toks:

B{'6vmdxpht5v73ea9fvp9t4v3x', false}

Srauto parametrai:

Sutarties sudarymo srautą sudaro tokie parametrai:

- *date_created* – sutarties sudarymo data,
- *contract* – sutarties tekstas.

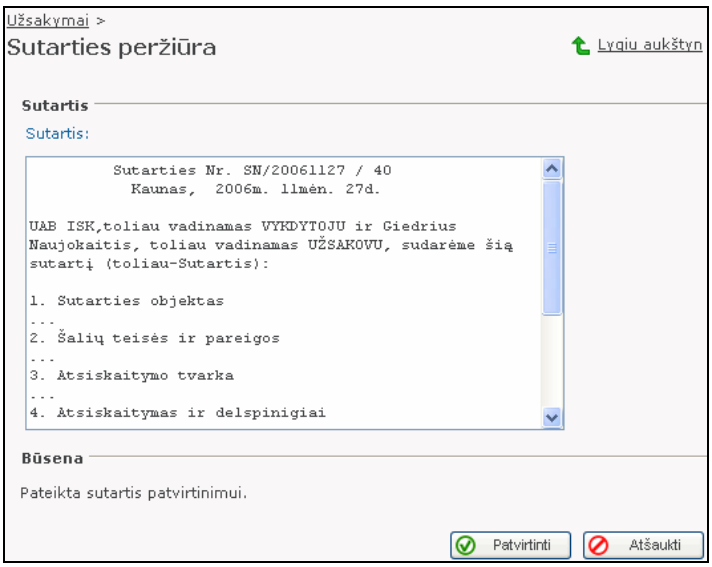
Naudojantis šiais parametrais sudaroma *Insert* užklausa, kuri įrašo naują sutartį į sutarčių lentelę (*contracts*).

Po būsenos parametrai:

Sudarius sutartį verslo procesas pereina į būseną „*Sutartis sudaryta*“. Būsenos parametrų sąrašą sudaro prieš būsenos parametrai ir srauto parametras *contract*:

B{'6vmdxpht5v73ea9fvp9t4v3x', false, 'Sutarties Nr. SN/20061...'}

- *Sutarties patvirtinimas* - sutarties įtvirtinimui reikalingas abiejų šalių sutikimas. Klientas turi patvirtinti, jog sutinka su sutarties sąlygomis. Patvirtinimo metu sutartis tampa pasirašyta. Sutarties patvirtinimo langas pateiktas *27 paveiksle*.



27 pav. Sutarties patvirtinimo langas

Prieš būsenos parametrai:

Sutarties patvirtinimo veiksmo prieš būsena sutampa su sutarties sudarymo po būsena.

Srauto parametrai:

Sutarties patvirtinimo srautą sudaro vienas parametras:

- *contract_confirmed* – nurodo sutarties patvirtinimą.

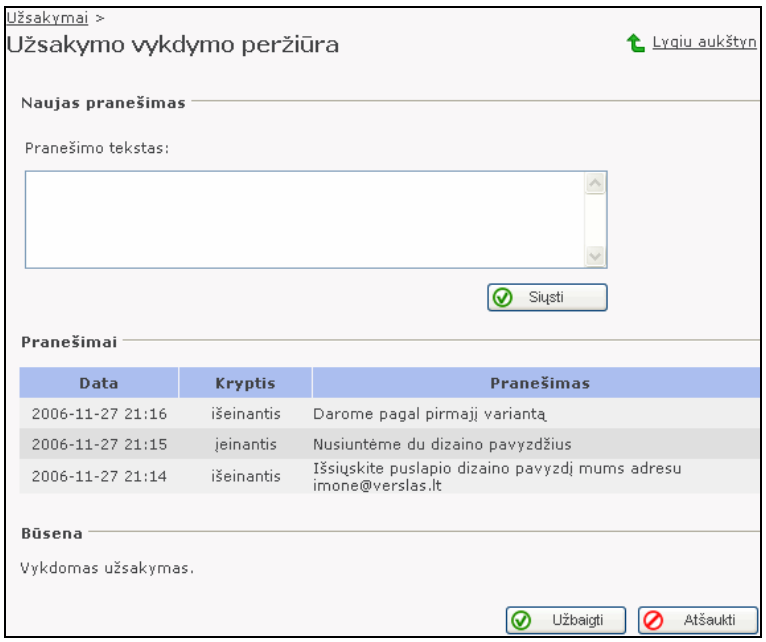
Naudojantis šiuo parametru sudaroma *Update* užklausa, kuri atnaujina sutarties būseną sutarčių lentelėje (*contracts*).

Po būsenos parametrai:

Patvirtinus sutartį verslo procesas pereina į būseną „*Sutartis pasirašyta*“. Būsenos parametrai išlieka tokie patys kaip ir prieš būsenoje, tik pasikeičia parametro *contract_confirmed* reikšmė:

B{'6vmdxpht5v73ea9fvp9t4v3x', true, 'Sutarties Nr. SN/20061...'}

- *Projekto vykdymas* – pasirašius sutartį, pradedamas projekto vykdymas. Projekto vykdymo metu yra laukiama, kol jis bus baigtas. Šio etapo metu įmonė gali bendrauti su klientu siųsdami vienas kitam pranešimus. Toks bendravimas leidžia lengviau pasiekti kliento tikslų, klientui netiesiogiai dalyvaujant kūrimo procese. Įmonės ir kliento bendravimo langas projekto vykdymo metu pateiktas *28 paveiksle*. Įmonei baigus vykdyti projektą, spaudžiamas užbaigimo mygtukas.



28 pav. Projekto vykdymo metu matomas langas

Prieš būsenos parametrai:

Prieš pradedant vykdyti projektą verslo procesas yra būsenoje „*Projektas nepradėtas*“. Būsenos parametrai:

- *order_number* – užsakymo numeris,
- *finished* – nurodo, ar projektas baigtas,
- *project_confirmed* – nurodo, ar projektas patvirtintas.

Būsena „*Projektas nepradėtas*“ susieta *exist* priklausomybe su būsena „*Sutartis pasirašyta*“. Tai reiškia, kad turi egzistuoti būseną „*Sutartis pasirašyta*“ aprašantis vektorius *B{'6vmdxpht5v73ea9fvp9t4v3x', true, 'Sutarties Nr. SN/20061...'}*, su kuriuo susijusi informacija jau yra įrašyta duomenų bazėje. *Exist* priklausomybės išlaikymui naudojamas parametras *order_number*.

Taigi projekto vykdymo prieš būsenos vektorius yra toks:

B{'6vmdxpht5v73ea9fvp9t4v3x', false, false}

Srauto parametrai:

Projekto vykdymo srautą sudaro tokie parametrai:

- *finished* – nurodo projekto užbaigimą,
- *messages* – projekto vykdymo metu siųsti pranešimai.
  - | - *date_entered* – pranešimo siuntimo data,
  - | - *author* – pranešimo autorius (įmonė ar klientas),

| - *message* – pranešimo tekstas.

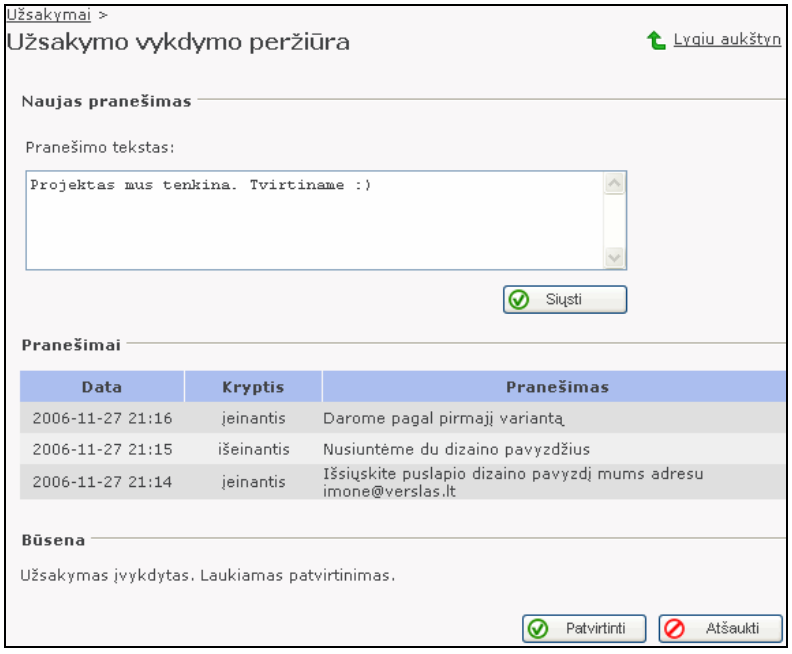
Naudojantis *finished* parametru sudaroma *Update* užklausa atnaujinanti užsakymo vykdymo būseną užsakymų lentelėje (*orders*). Naudojantis likusiais parametrais sudaromos *Insert* užklausos įrašančios siųstus pranešimus į pranešimų lentelę (*messages*). Pranešimų registravimas vyksta jų siuntimo metu prieš užbaigiant projekto vykdymą.

Po būsenos parametrai:

Baigus vykdyti projektą verslo procesas pereina į būseną „*Projektas baigtas*“. Būsenos parametrų sąrašą sudaro prieš būsenos parametrai su pasikeitusia parametro *finished* reikšme:

B{'6vmdxpht5v73ea9fvp9t4v3x', true, false}

- *Projekto įvertinimas* – užbaigus projektą, klientas peržiūri bei įvertina jį. Teigiamai įvertinus projektą jis tampa galutinai baigtas ir pereinama prie kito etapo. Projekto patvirtinimo langas pateiktas *29 paveiksle*.



29 pav. Projekto patvirtinimo langas

Prieš būsenos parametrai:

Projekto įvertinimo veiksmo prieš būsena sutampa su projekto vykdymo po būsena.

Srauto parametrai:

Projekto patvirtinimo srautą sudaro vienas parametras:

- *project_confirmed* – nurodo projekto patvirtinimą.

Naudojantis šiuo parametru sudaroma *Update* užklausa, kuri atnaujina projekto vykdymo būseną užsakymų lentelėje (*orders*).

Po būsenos parametrai:

Įvertinus ir patvirtinus projektą verslo procesas pereina į būseną „*Įvertintas projektas*“. Būsenos parametrų sąrašą sudaro prieš būsenos parametrai su pasikeitusia parametro *project_confirmed* reikšme:

B{'6vmdxpht5v73ea9fvp9t4v3x', true, true}

- *Mokėjimo reikalavimas* – galutinai baigus ir įvertinus projektą vykdomas projekto apmokėjimas. Mokėjimo reikalavimo metu išrašoma sąskaita už įvykdytą projektą. Sąskaita nusiunčiama klientui ir laukiamas apmokėjimas. Sąskaitos išrašymo langas pateiktas *30 paveiksle*.

30 pav. Sąskaitos formavimo langas

Prieš būsenos parametrai:

Prieš formuojant sąskaitą verslo procesas yra būsenoje „*Sąskaita neišrašyta*“. Būsenos parametrai:

- *order_number* – užsakymo numeris,
- *payed* – nurodo, ar sąskaita apmokėta,
- *peyment_received* – nurodo, ar gautas apmokėjimas.

Būsena „*Sąskaita neišrašyta*“ susieta *exist* priklausomybe su būsena „*Įvertintas projektas*“. Tai reiškia, kad turi egzistuoti būseną „*Įvertintas projektas*“ aprašantis vektorius

B{'6vmdxpht5v73ea9fvp9t4v3x', true, true}, su kuriuo susijusi informacija jau yra įrašyta duomenų bazėje. *Exist* priklausomybės išlaikymui naudojamas parametras *order_number*.

Taigi projekto vykdymo prieš būsenos vektorius yra toks:

B{'6vmdxpht5v73ea9fvp9t4v3x', false, false}

Srauto parametrai:

Mokėjimo reikalavimo srautą sudaro tokie parametrai:

- *date_entered*– nurodo sąskaitos suformavimo datą,
- *domain_cost* – domeno kaina (jeigu buvo reikalingas naujas domenas),
- *cms_cost* – turinio valdymo sistemos kaina (jeigu buvo reikalinga TVS),
- *php_cost* – PHP palaikymo kaina (jeigu buvo reikalinga PHP),
- *db_cost* – duomenų bazės kaina (jeigu buvo reikalinga DB),
- *realisation_cost* – projekto įvykdymo kaina,
- *total_cost* – bendra projekto kaina.

Naudojantis srauto parametrais sudaroma *Insert* užklausa įrašanti sąskaitą į sąskaitų lentelę (*invoices*).

Po būsenos parametrai:

Suformavus sąskaitą verslo procesas pereina į būseną „*Sąskaita išrašyta*“. Būsenos parametrų sąrašą sudaro prieš būsenos parametrai ir srauto parametrai:

B{'6vmdxpht5v73ea9fvp9t4v3x', fasle, false, '2006-11-28', 60, 400, 0, 0, 1000, 1460}

- *Apmokėjimas* – klientas gali peržiūrėti gautą sąskaitą (*31 paveikslas*). Norint atsispausdinti sąskaitą spaudžiama nuoroda „*Sąskaitos versija spausdinimui*“ ir atveriamas spausdinamos sąskaitos langas (*32 paveikslas*). Klientui apmokėjus sąskaitą spaudžiamas mygtukas „*Apmokėta*“. Tokiu būdu įmonei nusiunčiamas pranešimas apie apmokėjimo įvykdymą.

31 pav. Sąskaitos peržiūros langas

32 pav. Sąskaitos versija spausdinimui

Prieš būsenos parametrai:

Sąskaitos apmokėjimo veiksmo prieš būsena sutampa su mokėjimo reikalavimo (sąskaitos formavimo) po būsena.

Srauto parametrai:

Sąskaitos apmokėjimo srautą sudaro vienas parametras:

- *payed* – nurodo sąskaitos apmokėjimą.

Naudojantis šiuo parametru sudaroma *Update* užklausa, kuri atnaujina sąskaitos būseną sąskaitų lentelėje (*invoices*).

Po būsenos parametrai:

Apmokėjus sąskaitą verslo procesas pereina į būseną „*Sąskaita apmokėta*“. Būsenos parametrų sąrašą sudaro prieš būsenos parametrai su pasikeitusia parametro *payed* reikšme:

B{'6vmdxpht5v73ea9fvp9t4v3x', true, false, '2006-11-28', 60, 400, 0, 0, 1000, 1460}

- *Apmokėjimo patvirtinimas* – įmonė, gavusi pranešimą apie apmokėjimą, patikrina, ar apmokėjimas gautas. Kuomet gaunamas apmokėjimas, įmonė patvirtina apmokėjimo gavimą ir užsakymo vykdymas tampa galutinai užbaigtas. Apmokėjimo patvirtinimo langas pateiktas *33 paveiksle*.

33 pav. Sąskaitos apmokėjimo patvirtinimas

Prieš būsenos parametrai:

Apmokėjimo patvirtinimo veiksmo prieš būsena sutampa su apmokėjimo veiksmo po būsena.

Srauto parametrai:

Apmokėjimo gavimo patvirtinimo srautą sudaro vienas parametras:

- *peyment_received* – nurodo apmokėjimo gavimą.

Naudojantis šiuo parametru sudaroma *Update* užklausa, kuri atnaujina sąskaitos būseną sąskaitų lentelėje (*invoices*).

Po būsenos parametrai:

Patvirtinus sąskaitos apmokėjimą verslo procesas pereina į būseną „*Apmokėjimas gautas*“. Būsenos parametrų sąrašą sudaro prieš būsenos parametrai su pasikeitusia parametro *peyment_received* reikšme:

B{'6vmdxpht5v73ea9fvp9t4v3x', true, true, '2006-11-28', 60, 400, 0, 0, 1000, 1460}

Patvirtinus sąskaitos apmokėjimą, matomas užsakymo statistikos langas, kuris pateiktas *34 paveiksle*.

34 pav. Užsakymo statistikos langas

Pasibaigus paskutiniam internetinės svetainės kūrimo proceso etapui verslo procesas tampa galutinai užbaigtu, abiejų šalių tikslai pasiekti.

Jeigu internetinės svetainės kūrimo procesas buvo nutrauktas, tuomet vykdomas duomenų bazės transakcijų atstatymas. Pirmiausiai iš užklausų registro išrenkamos visos verslo proceso metu atliktos DB transakcijos. Išrinkimas vyksta pagal atributo *bursines_process_id* reikšmę, kuri eksperimentinėje sistemoje atitinka *order_id* reikšmę. Kiekvienai gautai užklausai sudaroma kompensacinė užklausa. Pavyzdžiui, pirmajai įvykdytai užklausai, kuri sukūrė užsakymą, sudaroma tokia kompensacinė užklausa:

DELETE FROM orders WHERE order_number = '6vmdxpht5v73ea9fvp9t4v3x'

Analogiškai sudaroma ir kitos kompensacinės *Delete*, *Update* bei *Insert* užklausos.

4.3. **Eksperimentinės sistemos architektūra**

Eksperimentui atlikti buvo naudojamos tokios priemonės:

a) MySQL duomenų bazė
b) PHP programavimo kalba

Sistema realizuoja verslo transakcijas replikuojančiose duomenų bazėse. Duomenų bazės replikavimui naudojamas replikavimas jungties pagrindu. Eksperimente naudojamos dvi duomenų bazės, kurių struktūros dalinai persidengia. Sistemos architektūrą vaizduoja *17 paveiksle* pateikta įdiegimo diagrama, kurioje atsispindi panaudotas tipinis trijų lygių architektūros modelis.

Replikuojančiose duomenų bazėse transakcijoms valdyti sukurtas specialus mechanizmas. Pagrindinės jo savybės bei privalumai:

- Atlieka verslo transakcijų bei jas atitinkančių duomenų bazės transakcijų sinchronizaciją replikuojančiose duomenų bazėse.
- Leidžia sinchronizuoti keletą viena po kitos sekančių DB transakcijų.
- Atlieka verslo transakcijų atstatymo sinchronizaciją replikuojančiose duomenų bazėse.
- Padeda išlaikyti replikuojančių duomenų bazių pilnumą bei nuoseklumą.
- Praneša vartotojui (įmonei ar klientui) apie ryšio nutrūkimą ar išjungtą kitos bendradarbiaujančios šalies serverį.

Duomenų bazės transakcijų kūrimui sukurtas automatizuotas užklausų sudarymo mechanizmas. Pagrindinės jo savybės bei privalumai:

- Leidžia sudaryti sintaksiškai tvarkingas įvairių tipų DB užklausas panaudojant mechanizmo įėjimo parametrus.
- Leidžia sudaryti sintaksiškai tvarkingas kompensacines užklausas, kurios naudojamos DB atstatymui.

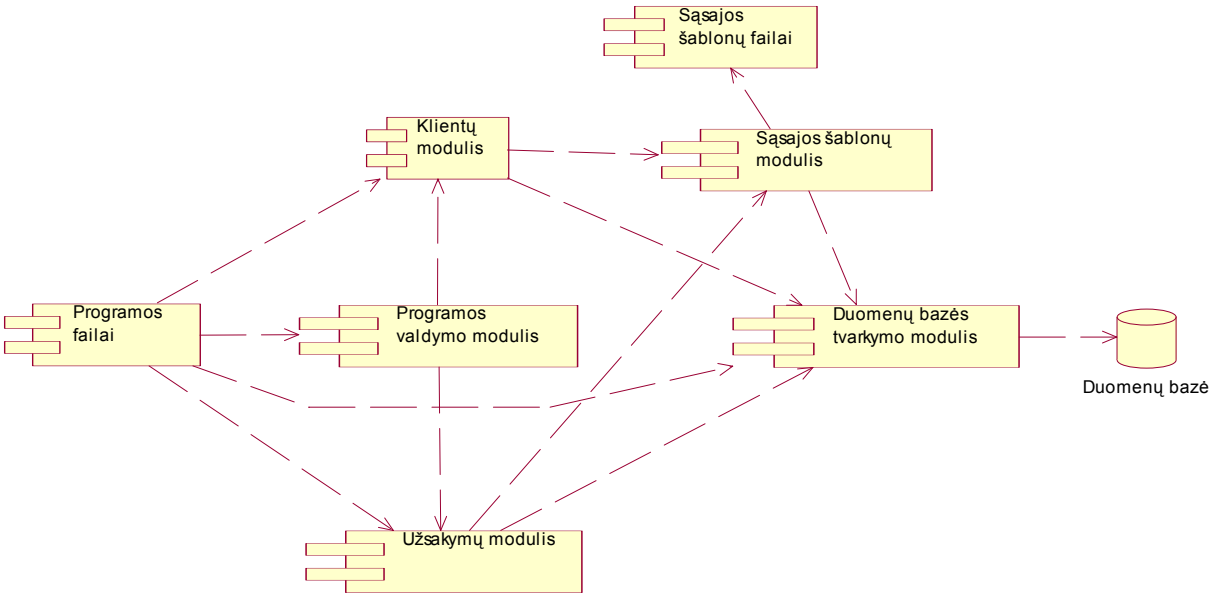
Kadangi eksperimente naudojamos dvi replikuojančios duomenų bazės, tuomet joms valdyti sukurtos dvi atskiros funkcionalumu panašios sistemos:

- *Klientų serverio sistema*. Atliekamos funkcijos:

- kliento registracija,
- paslaugų peržiūra bei jų užsisakymas
- užsakymo peržiūra bei dalyvavimas užsakymo vykdymo procese,
- asmeninių duomenų keitimas.

- *Įmonės sistema*. Atliekamos funkcijos:
  - klientų peržiūra bei registracijos patvirtinimas,
  - užsakymų peržiūra bei dalyvavimas užsakymo vykdymo procese,
  - įmonės duomenų keitimas.

Sistemos apimtį geriausiai pavaizduoja komponentų diagrama (*35 paveikslas*), kuri rodo fizinį sistemos vaizdą (komponentus bei jų tarpusavio priklausomybes). *35 paveiksle* pateikta komponentų diagrama yra tokia pati tiek klientų, tiek įmonės serverio sistemai, tačiau skirtinguose (įmonės bei kliento) serveriuose pavaizduoti komponentai atlieka skirtingas funkcijas.



35 pav. Sistemos komponentų diagrama

Duomenų bazės tvarkymo modulyje yra realizuoti užklausų sudarymo bei transakcijų sinchronizavimo replikuojančiose duomenų bazėse mechanizmai. Taigi duomenų bazės tvarkymo modulis jungiasi ne tik prie lokalios duomenų bazės, bet ir prie išorinės.

4.4. **Sistemos išplėtimo rekomendacijos**

Sukurta eksperimentinė sistema naudojama dvišaliam bendradarbiavimui. Verslo transakcijose dalyvauja viena įmonė ir jos klientai (atskirai kiekvienas klientas ir įmonė vykdo nepriklausomus verslo procesus). Klientų serveryje gali registruotis neribotas kiekis klientų ir kiekvienas jų gali užsisakyti įmonės teikiamas paslaugas.

Norint padidinti praktinį sistemos panaudojimą galima išplėsti sistemą, naudojantis tokiomis plėtros kryptimis:

- *Neribotas įmonių, teikiančių to pačio tipo paslaugas, kiekis*

  Galimybė prijungti neribotą kiekį įmonių yra realizuota esamoje eksperimentinėje sistemoje. Tereikia įdiegti programą (įmonei skirtą variantą) į prijungiamos įmonės serverį ir užregistruoti šią įmonę klientų serverio duomenų bazėje. Tokiu būdu klientai gali rinktis bet kurios įmonės teikiamas paslaugas.

- *Skirtingos klientų ir įmonių serverių duomenų struktūros*

  Esamoje eksperimentinėje sistemoje tiek kliento, tiek įmonės serverio duomenų struktūros persidengia, t.y. turi bendrų lentelių, kurios naudojamos verslo informacijai saugoti bei valdyti verslo transakcijas. Paprastai įmonė turi savo duomenų bazę, kurios dalies, naudojamos verslo informacijai saugoti, struktūra skiriasi nuo klientų serveryje naudojamos struktūros. Tokiu atveju atsiranda klasterių poreikis. Turi būti sudaromos sinonimų lentelės, kurios nurodys kokia lentelė ar jos atributas įmonės duomenų bazėje atitinka lentelę ar atributą klientų serverio duomenų bazėje. Sinonimų lentelė turėtų būti saugoma įmonės serveryje, nes kiekvienai įmonei ši lentelė bus skirtinga. Tokiu būdu galima valdyti duomenų srautus skirtingų struktūrų duomenų bazėse. Esant skirtingoms duomenų struktūroms tampa sudėtingesnis verslo transakcijų valdymas.

- *Skirtingo tipo paslaugas teikiančios įmonės*

  Prie esamos eksperimentinės sistemos galima prijungti tik įmones teikiančias to pačio tipo paslaugas. Norint prijungti naujo tipo teikiamas paslaugas reikėtų atlikti duomenų bazės struktūros atitikimo naujo tipo paslaugai analizę bei DB struktūros pertvarkymą. Tai sudėtingas nuo dalykinės srities priklausantis procesas, todėl plačiau neaptariamas.

Galima išskirti ir daugiau išplėtimo galimybių, tačiau paminėtos tik pagrindinės plėtros kryptys leisiančios padidinti praktinę sistemos naudą.

5. IŠVADOS

Norint garantuoti patikimą verslo informacinių sistemų funkcionavimą, jas projektuojant būtina formaliai aprašyti verslo transakcijas. Buvo apžvelgta keletas tokiam aprašymui siūlomų metodų. Remdamiesi apžvalga galime teigti, kad nėra vienareikšmiško elementarios verslo transakcijos apibrėžimo.

Buvo pastebėta, kad transakciją galima susieti su UML kalbos panaudojimo atvejo modeliu. Tada ji suprantama kaip aktoriaus veiksmų seka, duodanti tikslingą rezultatą. Tačiau toks traktavimas sukelia problemų dėl panaudojimo atvejo detalizavimo. Nėra būdų formaliai apibrėžti koks būtent veiksmas jau gali būti laikomas nedalomu. Tai visuomet priklauso nuo konkrečios situacijos.

Winograd - Flores darbų sekų modelis skirtas verslo transakcijoms specifikuoti yra nepriklausomas nuo konkrečios dalykinės srities. Be to, uždarų kilpų metodologija užtikrina, kad kiekvienas proceso žingsnis yra įvykdomas iki galo ir tiksliai, tokiu būdu vesdamas prie laukiamo galutinio rezultato. Tačiau Winograd - Flores modelis nėra pakankamas, nes jis modeliuoja tik dinaminę verslo transakcijų dalį.

Darbe pasiūlyta verslo transakcijas modeliuoti pragmatiškai motyvuotomis komunikacinėmis veiksmų kilpomis, kuriose įvesta duali veiksmo samprata, t.y. atvaizduojami ir srautai ir būsenos. EM metode įvykių sekų kontrolė yra sudėtinga. Ji įgyvendinama per statines priklausomybes (daugiausia kompozicinius ryšius). Todėl darbe panaudojamos gyvavimo ciklo (*exist*) priklausomybės, kurios įveda dinamiką ir supaprastina įvykių sekų kontrolę.

Binarinės veiksmų kilpos yra verslo transakcijų dinamikos šaltinis, iš kurių patogu realizuoti duomenų bazių transakcijas. Darbe pasiūlytas būdas formaliam verslo transakcijų aprašymui parenkant transakcijas nusakančius parametrus. Parametrai parenkami verslo transakcijų elementus skaidant į komponentines grupes, pogrupius ir t.t., kol informaciniai vienetai negali būti daugiau dalijami. Komponentiniai informacijos vienetai žemiausiame lygyje atitinka parametrus. Sukurtas automatinis užklausų sudarymo mechanizmas, kuris nepriklausomai nuo dalykinės srities gali sukurti DB užklausas panaudodamas gautus verslo transakcijų parametrus.

Kadangi verslo procesai bet kuriuo metu gali būti nutraukti, verslo transakcijas atitinkančios DB užklausos yra registruojamos darbe pasiūlytame įvykdytų užklausų žurnale. Esant sudarytai užklausų vykdymo istorijai, nutraukus verslo procesą, duomenų bazė sėkmingai atstatoma. Atstatymas atliekamas vykdant kompensacines užklausas pagal aprašytą DB atstatymo algoritmą.

Darbe nagrinėtos verslo transakcijos, kai verslo partneriai realizuoja savo servisus skirtingose aplinkose. Buvo naudojamos dvi replikuojančios duomenų bazės. Pasiūlyta naudoti jungtimis grįstą replikavimą. Tai padidina replikavimo sistemos lankstumą, leidžia kaupti mažiau duomenų, sistemą galima sėkmingai plėsti. Duomenų bazės transakcijų sinchronizacijai palaikyti pasiūlytas transakcijų sinchronizavimo mechanizmas, kuris užtikrina, kad DB transakcija įvykdoma abiejose duomenų bazėse arba, esant neprieinamam vienam iš serverių, nevykdoma nei vienoje duomenų bazėje.

Šiame darbe atliktame eksperimentiniame tyrime formaliai aprašyta internetinių puslapių kūrimo proceso sudėtinė komunikacinė kilpa. Praktiškai patikrinti užklausų sudarymo bei sinchronizavimo replikuojančiose duomenų bazėse mechanizmai. Išbandytas DB atstatymas po verslo proceso nutraukimo ir įsitikinta, kad vykdant DB transakcijas išlaikomas duomenų bazės pilnumas bei nuoseklumas.

6. LITERATŪRA

[1] Nemuraitė L. Elektroninio verslo procesų modeliavimo metodų tendencijos. Informacijos mokslai, Vilniaus universitetas, 2002.

[2] ebXML Business Specification Schema Version 1.01. Business Process Project Team, 2001. Prieiga per internetą: http://www.ebxml.org/specs/ebBPSS.pdf [žiūrėta 2005-10-12]

[3] ebXML Business Specification Schema Technical Specification v2.0.3. Committee Specification, 2006. Prieiga per internetą: http://docs.oasis-open.org/ebxml-bp/2.0.3/ebxmlbp-v2.0.3-Spec-cs-en-pdf.zip [žiūrėta 2006-09-13]

[4] Lina Nemuraitė, Bronius Paradauskas, Linas Salelionis. Extended communicative action loop for integration of new functional requirements. Kunas, 2002.

[5] Core Component Overview, ebXML Core Components. ebXML 2001.

[6] Rik Eshuis, Pierre Brimont, Eric Dubois, Bertrand Gr´egoire, Sophie Ramel. Animating ebXML Transactions with a Workflow Engine. Luxembourg, 2003.

[7] Dogac A., Yuruten B., Sapacapietra S. A Generalized Expert System for Database Design. IEEE Transactions on Software Engineering, 1989, 18(4), pp. 479-491.

[8] Thomas Strandenas, Randi Karlsen. Transaction Compensation in Web Services. Tromso 2003.

[9] Roger Wolter. Compensating Transactions. 2006. Prieiga per internetą: http://blogs.msdn.com/rogerwolterblog/archive/2006/05/24/606184.aspx [žiūrėta 2006-09-28]

[10] Walsh A. E. EbXML: the Technical reports. – Prentice Hall PTR, 2002

[11] ebXML Technical Architecture Specification v1.0.4. ebXML Technical Architecture Project Team, 2001-02-16. Prieiga per internetą: http://www.ebxml.org/specs/ebTA.doc [žiūrėta 2005-10-14]

[12] David D. Understanding ebXML, 2001-06-01. Prieiga per internetą: http://www-128.ibm.com/developerworks/ xml/library/x-ebxml/index.html [žiūrėta 2005-10-15]

[13] Michael C. Rawlins. ebXML - A Critical Analysis. Prieiga per internetą: http://www.rawlinsecconsulting.com/ ebXML/index.html [žiūrėta 2005-12-04]

[14] ebXML Core Component Dictionary, ebXML Core Components. Core Component Project Team, 2001m.

[15] Business Transaction protocol. Draft Specification, version 1.0. – Organization for the Advancement of Structured information Systems (OASIS), 2002.

[16] Thatte S. XLANG. Web Services for Business Process Design. Microsoft Corporation, 2001.

[17] Lina Nemuraitė, Bronius Paradauskas. From Use Cases to well structured conceptual schemas. Kaunas, 2006.

[18] Lina Nemuraitė, Bronius Paradauskas. Duomenų Bazės ir Semantiniai Modeliai. Kaunas, 2002.

[19] Remigijus Gustas. Semantic and Pragmatic Dependencies of Information Systems. Kaunas, 1997.
[20] Vytautas Brūzga. Duomenų Bazių Replikavimas ir Replikavimo Problemos. Kaunas, 2003.
[21] Raul Medina-Mora, Terry Winograd, Rodrigo Flores, Fernando Flores. The Action Workflow Approach to Workflow Management Technology. Action Technologies, Inc. 1301 Marina Village Parkway Alameda, 1992.
[22] Peter Fingar. Changing Change Itself. 2006.

7. TERMINŲ IR SANTRUMPŲ ŽODYNAS

Terminai:

- *verslo transakcija* – 1. verslo informacijos ir verslo signalų apsikeitimas tarp dviejų verslo partnerių, 2. darbo (veiksmų) tarp dviejų verslo partnerių vienetas, susidedantis iš vieno verslo partnerio inicijuojamo veiksmo ir kito verslo partnerio atsako.
- *duomenų bazės transakcija* – duomenų bazės SQL užklausa.
- *modelis* – realaus pasaulio objekto atvaizdavimas vienu ar kitu požiūriu.
- *serveris* – 1. informacinėse technologijose serveris – tai programa, kuri teikia paslaugas kitoms programoms bei sistemos vartotojams tame pačiame bei kituose kompiuteriuose, 2. kompiuteris, kuriame dirba serverio programa, paprastai yra vadinamas taip pat serveriu, 3. kliento/serverio programavimo modelyje serveris yra programa, kuri laukia ir įvykdo kliento programos užklausas.

Santrumpos:

- B2B – *Business to Business* (verslas - verslui)
- DB – *Database* (duomenų bazė)
- EM – *Enterprise Modeling* (verslo modeliavimas)
- BPSS – *Business Process Specification Schema* (verslo proceso specifikacijos schema)
- BPMN - *Business Process Modeling Notation* (verslo procesų modeliavimo notacija)
- CAL - *Communicative Action Loop* (komunikacinė veiksmų kilpa)
- SQL – *Structured Query Language* (struktūrizuota užklausų kalba)
- ebXML - *electronic business eXtensible Markup Language* (elektroninio verslo išplėstinė žymių kalba)
- XML - *eXtensible Markup Language* (išplėstinė žymių kalba)
- BTP – *business transaction protocol* (verslo transakcijos protokolas)
- UML – *Unified Modeling Language* (unifikuota modeliavimo kalba)
- HTTP – *Hypertext Transfer Protocol* (hiperteksto perdavimo protokolas)
- HTML – *Hypertext Markup Language* (hiperteksto užrašymo kalba)

8. PRIEDAI

Mokslinis straipsnis, paruoštas konferencijai Informacinės Technologijos 2007

VERSLO TRANSAKCIJŲ ATVAIZDAVIMAS DUOMENŲ BAZĖS TRANSAKCIJOMIS

Giedrius Naujokaitis, Bronius Paradauskas
Kauno Technologijos Universitetas, Informacijos sistemų katedra
Studentų 50, LT-3031 Kaunas, Lietuva

Šiame darbe apibrėžiama pragmatiškai motyvuota komunikacinė veiksmų kilpa, kurią pasiūlyta naudoti verslo transakcijoms modeliuoti ir duomenų bazių transakcijoms projektuoti. Komunikacinės veiksmų kilpų parametrais veiklos modelis adaptuojamas konkrečiai dalykinei sričiai. Parinkti parametrai panaudojami sudarant duomenų bazės transakcijas, orientuotas į veiklos ryšių objektų būsenų modifikavimą.

1. Įvadas

Elektroninės verslo transakcijos keičia tradicinius verslo procesus, nes siekiama pasiūlyti geresnes ir patogesnes paslaugas klientams ir partneriams, pamažinti procesų kaštus, juos darant operatyvesnius.

Informacijos sistemose:

- organizacijų darbų sekų valdymo standarto WfMC (angl. *Workflow Management Coalition*),
- verslas-klientui B2C (angl. *Business-To-Customer*)
- verslas-verslui B2B (angl. *Business-To-Business*)

vartotojų programos ir duomenys tendencingai atskiriami. Tokiose sistemose duomenų struktūros ir jų apdorojimo procesai nagrinėjami atskirai. Išorinių organizacijų verslo procesų transakcijų protokolu BTP (angl. *Business Transaction Protocol*) verslo transakcija apibrėžiama kaip nuosekli dviejų ar daugiau partnerių verslo ryšių būsenų kaita [1]. Verslo transakcija šaltinyje [2] apibrėžiama kaip verslo proceso dalis, vykdoma dviejų arba daugiau partnerių ir sugeneruojanti suskaičiuojamą galinę būseną (sėkmė arba nesėkmė).

Paskirstytiems verslo procesams būdingos ilgos trukmės (lyginant jas su duomenų bazių transakcijų trukmėmis), jie realizuojami, įvykdant daugelį DB transakcijų ir kaupiant duomenų srautus paskirstytosiose DB. BTP naudoja dviejų fazių bendradarbiavimo rezultatų skaičiavimo protokolą ir numato nedalomų darbų vienetų sujungimą į darnią verslo transakciją (derinį). Čia į sekų formavimo uždavinį leidžiama įsikišti iš išorės, patvirtinant arba atšaukiant nedalomus darbų vienetus. BTP užtikrina didelį verslo transakcijos dalyvių veiklos padarinių vertinimo ir patvirtinimo-atšaukimo įrankių lankstumą. Partneriai gali naudoti kompensacines operacijas, kad sugrįžti atgal arba judėti pirmyn ir tokiu būdu koordinuoti skaidomas verslo transakcijas. Verslo procesų koordinavimo tendencijos, panaudojant transakcijų sampratą, apžvelgtos [3]. Čia pastebėsime, kad neišbaigtos verslo transakcijos atšaukimas turi būti adekvačiai siejamas su DB atatinkamų užbaigtų transakcijų atšaukimu ir verslo ryšių pradinių būsenų atstatymu. Šis uždavinys tampa ypač sudėtingas, kai paskirstyti verslo partneriai eksploatuoja individualias nehomogenines DB, teikia paslaugų servisus iš skirtingų serverių.

Verslo transakcijų modelių lyginamoji analizė [4] rodo, kad verslo transakcija skirtingai suprantama skirtinguose elektroninio verslo modeliavimo metoduose ir neturi griežto formalaus apibrėžimo. Kai kuriuose protokoluose ar kalbose, pavyzdžiui, BTP arba XLANG [5] visai neapibrėžiama, kokios veiksmų sekos gali būti pripažįstamos transakcijomis. EbXML [6] bendradarbiavimas apibrėžiamas ekonominių įvykių pagrindu, tai yra, transakcijos reiškia ekonominius įvykius, kurių metu vienas partneris perduoda kitam ekonominę vertę ar ekonominių išteklių kontrolę. Šaltiniuose [7,8,9] verslo transakcijos specifikuojamos įprastinėmis priemonėmis (UML, XML). Tačiau tiesiogiai pritaikant šias priemones iškyla sunkumų [9] formaliai aprašant taisykles ir apribojimus, todėl tradicinių CASE įrankių panaudojimas yra ribotas. Darbe [10] verslo transakcija taip pat suprantama kaip universalios modeliavimo kalbos UML panaudojimo atvejis: panaudojimo atvejis yra veiksmų seka, kuri duoda tam tikrą reikšmingą rezultatą aktoriui, kuris šią seką inicijavo. Šio požiūrio pagrindinis privalumas yra tai, kad klasių diagrama (kartu su apribojimų kalba OCL specifikuotais kai kuriais papildymais) gali būti įdiegta objektinėse arba reliacinėse DB, tačiau UML kalbos panaudojimas verslo procesams specifikuoti dėl panaudojimo atvejų detalizavimo laipsnio neapibrėžtumo.

Peter Fingar savo paskutinės knygos [11] komentaruose [12] pateikia gamybos pakeitimų valdymo sistemos PCMS (angl. - Product Change Management System) principinius sprendimus, akcentuodamas verslo procesų turinio pirmumą prieš techninius sprendimus: „new business process management tools must go beyond the system-to-system and human-to-system paradigms, and on to *human-to-human interaction management*“. Verslo bendradarbiavimo modeliu BIM (angl. – Business Interaction Model) naudojamos dar 1983 m. Vinogrado-Flores aprašytos uždaros kilpos (Winograd-Flores Closed-Loop), jungiančios užsakovą (Customer) ir vykdytoją (Performer) į bendrą verslo ciklą, susidedantį iš keturių fazių:

1. užsakymo paruošimo (Preparation),

2. sąlygų aptarimo ir derybų – užsakymo priėmimo (Negatiation),
3. užsakymo vykdymo (Perfomance),
4. rezultatų priėmimas (Acceptance)

Winograd-Flores modelio pagrindinis privalumas yra tai, kad nei vienas verslo proceso žingsnis (kilpa) nėra traktuojamas kaip užbaigtas, kol atitinkamo žingsnio nepatvirtina klientas. Procesų projektuotojo žodžiais sakant, bendravimo modelis tarnauja kaip transliatorius verslo kalbą pakeičiantis technologijų kalba. Uždaro ciklo pritaikymo idėja išnagrinėta darbe [13], kur įmonės veiklos modeliu naudojama elementarių valdymo ciklų (EVC) darni visuma, apimanti tiek gamybos technologinius procesus, tiek valdymo informacinius procesus.

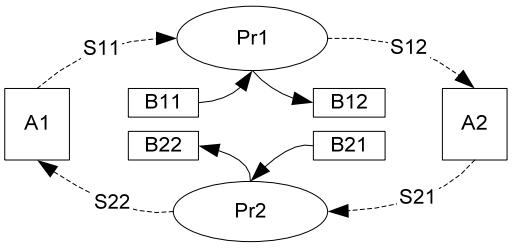
Darbe [14] pasiūlyta verslo transakcijas siūloma apibrėžti *pragmatiškai motyvuotomis* komunikacinių veiksmų kilpomis, kurios sudaro organizacijos modeliavimo EM (angl. Enterprise Modeling) metodo [15, 16] pagrindą. EM metodo idėja yra ta, kad organizacijos modeliu specifikuojami ne tik veikos *veiksmai*, bet ir pranešimų *srautai*. Kiekvienas veiksmas modelyje traktuojamas dvejopai: juo persiunčiamas *užsakovo* (aktoriaus A1) pranešimas *vykdytojui* (aktoriui A2) ir jis keičia verslo objektų būseną iš *prieš-būsenos* B11 į *po-būseną* B12. Darbe [14] priimta prielaida, kad pranešimų srautai yra *asinchroniniai*, t. y. aktoriaus A1 pasiųstas duomenų srautas S11 gali nesutapti su srautu S12, kurį priima aktorius A2, ir šie srautai nėra laiko požiūriu sinchronizuoti.

Atlikus verslo transakcijų valdymo analizę nustatyta, kad naudoti tradicinį atominį transakcijų valdymą yra neefetyvu dėl resursų blokavimo, kai transakcijas reikia vykdyti paskirstytose sistemose. Šio darbo tikslas yra parengti metodinius principus ilgos trumės sudėtinių verslo transakcijų projektavimui.

2. Verslo transakcijų vaizdavimas komunikacinėmis veiksmų kilpomis su statinėmis ir dinaminėmis duomenų priklausomybėmis

Patogu verslo transakcijas modeliuoti naudojant komunikacines veiksmų kilpas [14], kuriomis galima modeliuoti verslo transakcijų statiką, dinamiką bei pragmatiką. Verslo transakcijos apibrėžiamos kaip komunikacinė veiksmų kilpa (angl.- *Communicative Action Loop* arba *CAL*). Bendru atveju verslo transakcija gali būti sulyginta su panaudojimo atveju, kaip eilė atliekamų veiksmų, duodančių tam tikrą rezultatą verslo partneriui (aktoriui), kuris tuos veiksmus inicijavo. Verslo transakciją sudaro veiksmai sujungti į uždarą kilpą. Jeigu vis dar laukiame verslo partnerio atsako, tai verslo transakcija dar neužbaigta [14].

1 pav. pateikta komunikacinė (binarinė) veiksmų kilpa sudaryta iš dviejų aktorių – kliento (A1) ir atlikėjo (A2), pranešimų srautų S11÷S22, ir dviejų veiksmų – kliento veiksmo (Pr1) ir atlikėjo veiksmo (Pr2). Dinaminiai situacijos pasikeitimai prieš ir po veiksmų pateikiami kaip perėjimai nuo būsenų B11, B21 į būsenas B12, B22 ir aibė pragmatinių bei semantinių priklausomybių tarp jų.



1 pav. Bazinė komunikacinė veiksmų kilpa

Tam tikros srities versle vykdomos transakcijos yra tarpusavyje susijusios. Kiekviena transakcija vienaip ar kitaip keičia verslo santykių būseną tarp bendradarbiaujančių partnerių. Verslo transakcijų metu bendradarbiaujančios šalys keičiasi verslo informacija (pranešimais), kuri vis papildo vykstantį verslo procesą ir veda jį į užbaigimo būseną atitinkančią verslo partnerių siekiamus tikslus.

Bendru atveju verslo transakcijos veiksmas turi *prieš* ir *po* būseną. Elementari binarinė kilpa susideda iš dviejų veiksmų, kurių kiekvienas turi *prieš*- ir *po*- būsenas [16] (iš viso 4 būsenos). Pirmojo transakcijos veiksmo *po*- būsena gali sutapti su antrojo veiksmo *prieš*- būsena (**2a pav.** būsenos *B1* ir *B4*). Būsenos gali turėti duomenų komponentus (**2b pav.**). Panaudotame veiklos modelyje tarp duomenų statinių ir dinaminių priklausomybių nustatytos aštuonios pagrindinės prielaidos ir išvedimo taisyklės. Sakykime, aktorius A1 pranešimu S11 inicijavo sistemos panaudojimo atveją Pr1, o recipijentas A2 atsakė pranešimu S21, realizuojant sistemos panaudojimo atveją Pr2 (žr. **2 pav.**).

1) Jeigu veiksmas Pr1 *prieš*- būsenos neturėjo, tai *po*-būsena B1 sutampa su duomenų srautu, t.y. S11=S12=B1;
2) *Po*- būsena B2 po veiksmo Pr2 paskaičiuojama, kaip būsenos B1 ir srauto S2 jungtis [17]; analogiškai būsena B4 nustatoma iš būsenos B3 ir srauto S31;
3) Jeigu paskutinėje interakcijoje po veiksmo Pr4 po- būsena diagramoje nevaizduojama, tai ji sutapatinama su srautu S42;
4) Asinchroninių pranešimų atveju siunčiamas srautas S21 gali nesutapti su priimamu srautu S22; analogiškai S32 gali nesutapti su S31;

5) Po- būsena B2 visada yra dinaminėje gyvavimo ciklo priklausomybėje nuo prieš- būsenos B1, t.y. B2*exist*B1;

6) Jeigu tos pačios verslo transakcijos dvi būsenos, pavyzdžiui, B2 ir B1 yra sujungtos statine poaibio priklausomybe B2 C B1, tai tada teisinga yra ir dinaminė priklausomybė B2*exist*B1; atvirkščias teiginys yra neteisingas;

7) Tos pačios verslo transakcijos vienos būsenos, pavyzdžiui, B1 du duomenų komponentai K11 ir K12 visada yra dinaminėje *sambūvio* priklausomybėje K11*coexist*K12, t.y.:

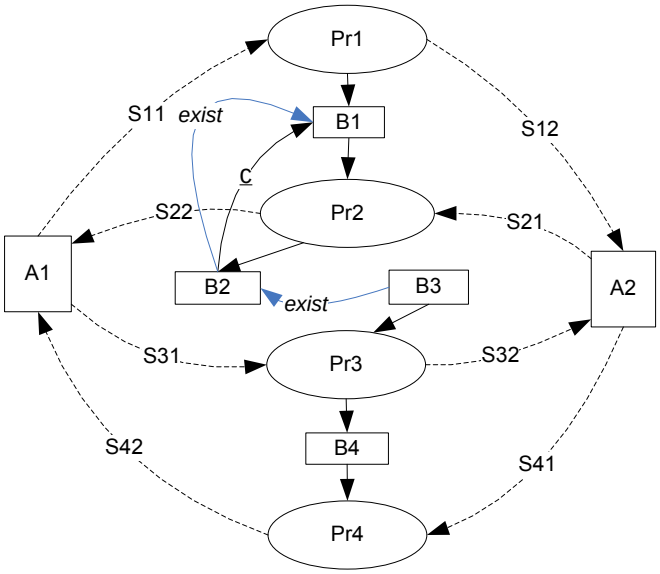
iš B1*composed_of* K11 ir B1*composed_of* K12
seka B1.K11 *coexist* B1.K12;

tokie du komponentai būtinai turi bent vieną *tėvinį* objektą T, kuris panaudojamas *join* operacijoje [17], kaip jungimo kriterijus;

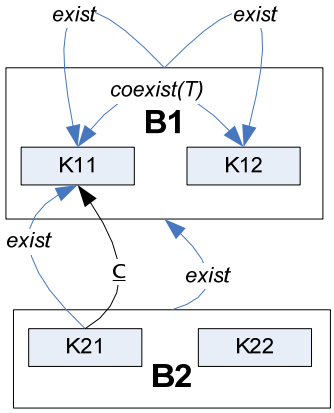
8) Jeigu būsenos B1 komponentas K11 ir būsenos B2 komponentas K21 susieti poaibio priklausomybe B2.K21 C B1.K11, tai iš jos seka priklausomybė B2.K21*exist* B1.K11;
iš priklausomybės B2.K21exist B1.K11 seka B2*exist* B1;

šios dvi taisyklės rekursyviai nustato trečiąją:

iš B2.K21 C B1.K11 seka B2*exist* B1



a) būsenos B1, B2, B3, B4 yra paprastosios (vienkomponentinės)



b) būsenos B1 ir B2 yra sudėtinės, turinčios duomenų komponentaus K11, K12 ir K21, K22

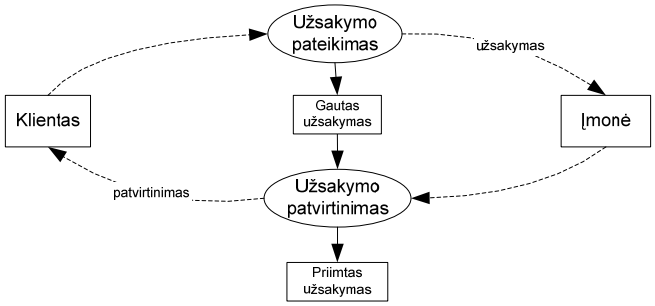
2 pav. Dinaminės būsenų gyvavimo (*exist, coexist*) ir statinės duomenų *(kompozicijos, poaibio*) priklausomybės

Dinaminės *exist* ir *coexist* priklausomybės, išvedamos iš statinių *poaibio* ir *kompozicijos* priklausomybių, gali būti nespecifikuojamos ir nustatomos nutylint. Paskutinioji (aštuntoji) priklausomybė svarbi sudėtinių verslo transakcijų dinaminei integracijai, kai viena verslo transakcija realizuojama keliomis binarinėmis kilpomis.

3. Komunikacinių veiksmų kilpų parametrų parinkimas

Atlikus verslo transakcijų modeliavimą gaunamas specifikuotas verslo proceso modelis. Gautas modelis gali būti panaudotas informacinės sistemos, vykdančios sumodeliuotas transakcijas, kūrimui. Tačiau norint kompiuterizuoti verslo transakcijas, būtina komunikacines veiksmų kilpas detaliai išanalizuoti identifikuojant su transakcija susijusių aktorių, veiksmų, informacijos srautų bei būsenų informaciją, kuri reikalinga sistemai funkcionuoti. Tokiu būdu dekomponuojami verslo transakcijų komponentai iki žemiausio lygio, kuriame yra išvardijami verslo transakcijas nusakantys parametrai. Kitaip sakant vykdomas transakcijų parametrų, saugančių verslo informaciją, parinkimas. Vykstant verslo transakcijoms verslo partneriai keičiasi informacija, kuri aprašoma parametrų rinkiniais. Turėdami aiškiai apibrėžtą informaciją galime ją saugoti duomenų bazėje, nes parametrai nusako duomenų struktūrą reikalingą tai informacijai išsaugoti.

Nustatant verslo proceso parametrus pirmiausia reikalingas detalus verslo proceso modelis. Modelis turi būti pakankamas reikalingos verslo informacijos identifikavimui, pavyzdžiui, turime modelį, kurio dalis (**3 pav.**) vaizduoja pirkėjo reikalavimų siuntimą norimai paslaugai gauti.

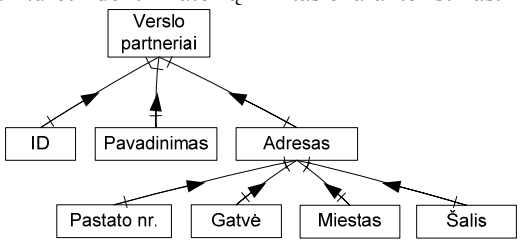


3 pav. Komunikacinės veiksmų kilpos pavyzdys

Naudojantis **3 pav.** pavaizduota kilpa galima išskirti reikalingą verslo informaciją. Galima naudotis tokiais žingsniais [1]:

- Padalinti verslo informaciją į komponentines grupes ir joms suteikti pavadinimus. Pavyzdžiui:
  - *Verslo partneriai*
  - *Užsakymo informacija*
  - *Reikalavimai užsakymo vykdymui*

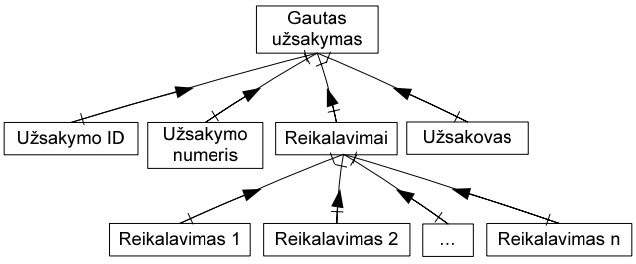
- Kiekvieną komponentinę grupę dalinti į mažesnius komponentinius vienetus (**4 pav.**). Kiekviena komponentinė grupė gali turėti identifikatorių ir kitas charakteristikas. Pavyzdžiui:



4 pav. Informacijos komponentinės struktūros pavyzdys

- Tęsti komponentinį dalijimą toliau kol pasiekiamas žemiausias lygis, kuomet informacija jau negali būti padalijama.
- Komponentiniai informacijos vienetai žemiausiame lygyje atitinka parametrus, kurie naudojami vykstant verslo transakcijai.

Parametrų rinkiniai naudojami ne tik srautų informacijos, bet ir būsenų aprašymui. Pavyzdžiui, **5 pav.** pateiktas būseną *gautas užsakymas* nusakančių parametrų rinkinio fragmentas.



5 pav. Būsena nusakančių parametrų rinkinio fragmentas

Vykstant transakcijai parametrų kiekis ir jų reikšmės keičiasi. Esant tokiems pasikeitimams keičiasi ir verslo proceso būsena. Saugant būsenas nusakančių parametrų reikšmes yra valdomas verslo procesas. Gyvavimo ir poaibio būsenų priklausomybės realizuojamos susietų būsenų bendrais parametrų rinkiniais (bendrais komponentais, **2 pav.**).

4. Duomenų bazės transakcijų sudarymas

Duomenų bazės transakcijų sudarymas - tai perėjimas nuo verslo transakcijų prie DB transakcijų. Vizualiai pateiktas verslo procesas paverčiamas konkrečiomis DB užklausomis. Parametrizuotos komunikacinės veiksmų kilpos yra pagrindas duomenų bazės transakcijų sudarymui.

Naudojantis komunikacinių kilpų parametrais automatiškai sudaromos DB užklausos (panaudojant automatizuotą užklausų sudarymo įrankį). Sudarant užklausas išskiriami trys DB modifikavimo užklausų tipai: *insert*, *update* ir *delete*. Šių užklausų atlikimui gali būti reikalinga ir duomenų išrinkimo užklausa *select*, kurios rezultatas gali būti panaudojamas modifikavimo užklausose kaip vienas iš parametrų. Parametrų panaudojimas sudarant užklausas pateiktas *1 lentelėje*.

1 lentelė. Komunikacinių kilpų parametrų panaudojimas sudarant užklausas

Užklausos tipas	Parametrų panaudojimas
Insert	verslo transakcijos parametrai ar jų dalis naudojami kaip užklausos DB lentelių atributų pavadinimai, o parametrų reikšmės naudojamos kaip įterpiami duomenys
Update	verslo transakcijos parametrai ar jų dalis naudojami kaip užklausos DB lentelių atributų pavadinimai, o parametrų reikšmės – kaip atnaujinami duomenys. Parametrai su reikšmėmis gali būti panaudojami ir užklausos filtrui
Delete	verslo transakcijos parametrai, atitinkantys DB laukų pavadinimus, ir jų reikšmės naudojami užklausos filtrui
Select	verslo transakcijos parametrai ar jų dalis naudojami kaip užklausos DB lentelių atributų pavadinimai, kurių reikšmės išrenkamos. Parametrai su reikšmėmis gali būti panaudojami ir užklausos filtrui

Sudaryta užklausa turi būti sintaksiškai tvarkinga. Užklausos tvarkingą sudarymą turi užtikrinti automatizuotas užklausų sudarymo įrankis.

5. Išvados

Norint garantuoti patikimą verslo informacinių sistemų funkcionavimą, būtina formaliai aprašyti verslo transakcijas. Projektuojant verslo valdymo sistemas, verslo transakcijoms specifikuoti tikslinga naudoti komunikacines veiksmų kilpas. Uždarų kilpų metodologija užtikrina, kad kiekvienas proceso žingsnis yra įvykdomas iki galo ir tiksliai, tokiu būdu vesdamas prie laukiamo galutinio rezultato.

Komunikacinių kilpų pagrindu parinkus verslo transakcijų parametrus, sudaromos duomenų bazės transakcijos. DB transakcijų sudarymui turi būti naudojamas automatizuotas užklausų sudarymo mechanizmas.

Literatūros sąrašas

[1]. Business Transaction Protocol. Draft Specification, version 1.0. – Organization for Advancement of Structured Information Systems (OASIS), 18 March 2002.

[2]. UN/CEFACT and OASIS. ebXML Glossary. Interneto prieiga: www.ebxml.org/specs/ebGLOSS.pdf, 2001

[3] (Dayal, 2001). Dayal U., Hsu M., Ladin R. Business Process Coordination: State of Art, Trends and open Issues. – Proceedings of the 27th International Conference of Very Large Data Bases. – Roma, Italy, September 11-14, 2001, pp. 3-13.

[4]. Nemuraitė L. Elektroninio verslo procesų modeliavimo metodų tendensijos / Informacijos mokslai. Vilniaus universitetas, 2002, No. 21, p. 77-88.

[5]. Thatte S. XLANG. Web Services for Business Process Design. Microsoft Corporation, 2001.

[6]. Walsh A. E. (Ed.) EbXML: the Technical Reports. – Prentice Hall PTR, 2002.

[7]. Bertrand G., Incoul Ch., Ramel S., Schmitt M., Gautheron L., Brimont P., Dubois E. Efficient: A Framework for Animating and Validating e-Business Transactions. Interneto prieiga: http://www.ercim.org/publication/Ercim_News/enw58/incoul.html, 2004.

[8]. Hofreiter B., Huemer Ch., Zapletal M. Registering UMM Business Collaboration Models in an ebXML Registry. Interneto prieiga: http://www.cs.univie.ac.at/upload//shared/mis/publications/IEEE-CEC06-final.pdf, 2006.

[9]. Eshuis R., Brimont P., Dubois E., Gregoire B., Ramel S. Animating ebXML Transactions with a Workflow Engine. Interneto prieiga: http://efficient.citi.tudor.lu/pub/Animating_ebXML_transactions_with_a_workflow_engine_EFFICIENT_CoopIS03.pdf, 2003.

[10]. Nemuraitė L., Paradauskas B. From Use Cases to Well Structured Conceptual Schemas, in: Vasilecas, O. et all (Eds.), Information Systems Development. Advances in Theory, Practice, and Education. Springer, 2005, XXVIII, ISBN: 0-387-25026-3, p. 303-314.

[11] Fingar P. Extreme Competition: Innovation and the Great 21st Century Business Reformation, Meghan-Kiffer Press, 2006.

[12]. Finger P. Changing Change Itself. Interneto prieiga: http://www.bptrends.com/publicationfiles/06-06COL-ChangingChangeItself-Fingar1.pdf, 2007

[13]. Gudas S., Lopata A., Skersys T. Approach to Enterprise Modelling for Information Systems Engineering. INFORMATICA, Vol. 16, No. 2, Institute of Mathematics and Informatics, Vilnius, 2005, pp. 175-192.

[14]. Nemuraitė L., Paradauskas B., Salelionis L. Extended Communicative Action Loop for Integration of New Functional Requirements // Information technology and control. ISSN 1392-124X. Kaunas: Technologija, 2002, No. 2(23), p. 18 - 26.
[15]. Gustas R. Towards a Communication Based Approach for Enterprise Modelling Integration /International Conference on Integration and Reuse, Honolulu, Hawaii, USA, November 1-3, 2000.
[16]. Gustas R. Semantic and pragmatic dependencies of information systems.- Monograph, Technologija, 1997, pp. 274.
[17]. Paradauskas B., Nemuraitė L. Duomenų bazės ir semantiniai modeliai. – Monografija, Technologija, 2002, 264 p.

REPRESENTING BUSINESS TRANSACTIONS AS DATABASE TRANSACTIONS

In this paper the pragmatically motivated communicative action loop (*CAL*) is proposed for modeling business transactions and designing database transactions. Using parameters of communicative action loops, activity model can be adapted to particular objective area. Selected parameters are utilizable in formation of database transactions that are oriented in modification of states of objects of activity relations. Mechanism of domain calculating logic is used to simulate alternation of states.